Universidade de Aveiro Departamento de Comunicação e Arte
2014

Marisa Alexandra Tavares Martins

SELEÇÃO DE UMA *FRAMEWORK* MVC CLIENT-SIDE PARA UMA APLICAÇÃO WEB E MOBILE

Universidade de Aveiro Departamento de Comunicação e Arte
Ano 2014

Marisa Alexandra Tavares Martins

SELEÇÃO DE UMA *FRAMEWORK* MVC CLIENT-SIDE PARA UMA APLICAÇÃO WEB E MOBILE

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Comunicação Multimédia, realizada sob a orientação científica do Professor Doutor Pedro Beça do Departamento de Comunicação e Arte da Universidade de Aveiro

Dedico esta dissertação aos meus pais e irmão.

o júri

presidente

Prof. Doutora Ana Isabel Barreto Furtado Franco de Albuquerque Veloso
professora auxiliar no Departamento de Comunicação e Arte da Universidade de Aveiro

Prof. Doutor Diogo Nuno Pereira Gomes
professor auxiliar no Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutor Pedro Miguel dos Santos Beça Pereira
professor auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro

agradecimentos

Quero agradecer à minha família e amigos por toda a paciência e apoio que me concederam nesta longa fase. Agradeço também ao meu orientador Professor Pedro Beça pela disponibilidade e ajuda ao longo do desenvolvimento deste trabalho e pelo esforço de me manter sempre focada e na direção correta aos objetivos desta dissertação. Quero agradecer também ao meu co orientador e CEO da empresa Prodcent Engenheiro Pedro Sousa Rêgo, pela experiência e longa aprendizagem que obtive ao realizar a minha investigação em contexto empresarial na entidade descrita. Aos meus colegas de trabalho que me acompanham desde o início desta dissertação quero agradecer pela disponibilidade de me ouvirem e ajudarem nas entrevistas finais para obter os últimos resultados e conclusões. Por fim e também importante destacar, quero agradecer ao Mário Moreira, trabalhador na empresa PT Inovações, ao Ricardo Ferrolho e Luís Almeida, trabalhadores na empresa Ubiwhere, ao Professor Carlos Santos e ao Professor Nuno Ribeiro pela disponibilidade ao longo do processo de recolha de dados para a investigação, que permitiu dados bastante completos para prosseguir o meu estudo.

palavras-chave

Aplicações web; *business-to-business*; *front-end*; *Frameworks mvc javascript*; *mobile*;

resumo

Framework web é um conceito cada vez mais comum na atualidade devido à sua capacidade de permitir facilitar e ajudar programadores e gestores de empresas na estruturação e manutenção do seu código. Atualmente existem diversas *Frameworks* disponíveis, através das quais é possível tornar mais eficaz a organização e delimitação das várias funcionalidades e fases da programação de aplicações web. Devido há existência de diversas *Frameworks*, algumas das quais similares, o processo de escolha da *Framework* ideal pode ser complexo. Neste processo, é importante ter em consideração diversos fatores que vão ao encontro dos objetivos do projeto que se pretende desenvolver, bem como da empresa e da equipa de desenvolvimento. Desta forma, neste documento procura-se definir e caracterizar os melhores critérios para a seleção de uma *Framework* mvc javascript para aplicações web e móveis.

keywords Web applications; business-to-business; front-end; Frameworks mvc javascript; mobile;

abstract The *Framework* web is a concept that has become increasingly known by virtue of its ability to facilitate and help programmers and business managers when it comes to structuring and managing code. Nowadays there are several *Framework*s available through which it is possible to improve the efficiency of the organization and definition of multiplicity of functionalities and programming stages of web applications. Due to the wide range of *Framework*s available, most of them very similar, the process of choosing the ideal *Framework* becomes much more complex. Throughout this process, it is crucial to take into account several factors that meet the objectives of the project that is going to be developed, as well as those of the company and the development team. Therefore, this document seeks to define and characterize the best selection criteria for a mvc javascript *Framework* for web and mobile applications.

ÍNDICE

I. INTRODUÇÃO E ENQUADRAMENTO DA INVESTIGAÇÃO 1

A. CONTEXTUALIZAÇÃO 1
B. PERGUNTA DE INVESTIGAÇÃO 2
C. FINALIDADES E OBJETIVOS 3
D. ESTRUTURA DA DISSERTAÇÃO 4

II. ENQUADRAMENTO TEÓRICO 5

A. APLICAÇÕES E *FRAMEWORK*S WEB PARA *BUSINESS*-TO-*BUSINESS* (B2B) 5
1. B2B 6
2. Aplicações web Front-end (cliente) Client-side 8
3. Frameworks web 10
a) Javascript MV* e MVC 12
b) *Framework*s móveis 20
c) *Framework*s CSS 25
B. CRITÉRIOS PARA ESCOLHA DA *FRAMEWORK* (MV*) 26
1. Ferramentas e técnicas para auxiliar a seleção de uma Framework 27
2. Critérios gerais 28
3. Compatibilidade 30
4. Responsividade 32
5. Performance da aplicação 34
a) Algumas ferramentas para auxiliar na performance 35
6. Economia de custos de produção 35

III. METODOLOGIAS DE INVESTIGAÇÃO 37

A. METODOLOGIA ESCOLHIDA 37
B. MODELO DE ANÁLISE 39
C. RECOLHA DE DADOS 40
D. ETAPAS DO ESTUDO 42

IV. RECOLHA E ANÁLISE DOS DADOS 45

A. ANÁLISE DAS ENTREVISTAS 45
B. ANÁLISE DOS DADOS 48

V. PROJETO EM CONTEXTO EMPRESARIAL 55

A. DESCRIÇÃO DA EMPRESA E DO PROJETO 55

B. PROCESSO DE ESCOLHA DA *FRAMEWORK* JAVASCRIPT NA EMPRESA 58
1. Experiência e satisfação com a Framework escolhida 63

VI. AVALIAÇÃO DAS *FRAMEWORK*S MENCIONADAS 69

VII. CAMINHOS DE DECISÃO 73

A. CRITÉRIOS DE ESCOLHA PRINCIPAIS 73
B. ESCOLHA DA *FRAMEWORK* FINAL 75

VIII. VALIDAÇÃO DA *FRAMEWORK* ESCOLHIDA 83

A. REQUISITOS DO PROJETO 83
B. RECOLHA E ANÁLISE DE DADOS 87
1. Análise da entrevista 87
2. Análise dos dados 89

IX. CONCLUSÕES 95

A. LIMITAÇÕES 96
B. TRABALHO FUTURO 96

X. BIBLIOGRAFIA 99

XI. ANEXOS 103

A. ANEXO 1 - GUIÃO DE ENTREVISTA 103
B. ANEXO 2 – RESUMO DA ANÁLISE DOS DADOS DA PRIMEIRA ENTREVISTA 104
C. ANEXO 3 – ONE WAY DATA BINDING E TWO WAY DATA BINDING 105
D. ANEXO 4 – GUIÃO DE ENTREVISTA AOS ELEMENTOS DA EMPRESA 106

ÍNDICE DE ILUSTRAÇÕES

Ilustração 1- Model-View-Controller 13
Ilustração 2 - Dispositivos e sistemas operativos móveis 20
Ilustração 3 - diferentes resoluções e orientação dos dispositivos 33
Ilustração 4 - Etapas do estudo 43
Ilustração 5 - Trending de algumas *Framework*s JavaScript MVC na Google 59
Ilustração 6 - Processo de publicação de um catálogo - catálogo não publicado 64
Ilustração 7 - Processo de publicação de um catálogo - catálogo publicado 64
Ilustração 8 - Validações de formulário - AngularJs 66
Ilustração 9 - Interesse global em resultados de pesquisa 78
Ilustração 10 - Interesse global por vídeos no Youtube 79

ÍNDICE DE TABELAS

Tabela 1 - Modelo de Análise 39
Tabela 2 - Categorização dos entrevistados 46
Tabela 3 - Lista dos principais critérios utilizados para ajudar na decisão pela *Framework* final 51
Tabela 4 - *Framework*s e modelo mvc 59
Tabela 5 - Resultados dos critérios analisados 61
Tabela 6 - *Frameworks* mencionadas e frequência 71
Tabela 7 - Resultados das *Frameworks* vs. Critérios 75
Tabela 8 - Resultados das *Frameworks* vs. critérios 76
Tabela 9 - *Framework*s vs. requisitos do projeto 84
Tabela 10 - *Frameworks* vs. requisitos do projeto 85
Tabela 11 - caracterização das fontes de informação 88
Tabela 12 - Pontos positivos e negativos da *Framework* Angularjs 91

GLOSSÁRIO

e-commerce – *electronic commerce*
B2B – *Business-to-Business*
e-marketplaces – *electronic marketplaces*
e-procurement – *electronic procurement*
e-distribution – *electronic distribution*
HTML – *Hypertext Markup Language*
CSS – *Cascading Style Sheets*
API – *Aplication Programming Interface*
JSON – *Javascript Object Notation*
XML – *eXtensible Markup Language*
SPAs – *Single Page Application*
DOM – *Document Object Model*
MVC – *Model View Controller*
MVP – *Model View Presenter*
MVVM – *Model View View Model*
MV* - *Model View**
HTTP – *Hypertext Transfer Protocol*
CRUD – *Create Read Update Delete*
DI – *Dependency Injection*
MVW – *Model View Whatever*
YUI – *Yahoo User Interface*
SDKs - *Software Development Kits*
PE – *Progressive Enhancement*
REST - *Representational State Transfer*
SOAP - *Simple Object Access Protocol*
SQL - *Structured Query Language*
W3C - *World Wide Consortium*
WSDL - *Web Service Description Language*
SEO – *Search Engine Optimization*
UX – *User Experience*

I. INTRODUÇÃO E ENQUADRAMENTO DA INVESTIGAÇÃO

A. Contextualização

O conceito de Web 2.0 retrata um conjunto de tendências económicas, sociais e tecnológicas que coletivamente criam as bases para a uma madura e distinta internet caracterizada pela interação do utilizador, abertura e efeitos de rede (Musser, s.d). Com o aparecimento da Web 2.0 verificaram-se alterações nas empresas ao nível da estratégia de realização de negócios (Musser, s.d.). As aplicações de contacto com os clientes tornaram-se mais dinâmicas e acessíveis (Musser, s.d). O leque alargado de ferramentas e tecnologias disponíveis atualmente, que permitem criar todo este dinamismo e acessibilidade e que facilitam e auxiliam os programadores, aumentam a dificuldade no iniciar de um novo projeto nesta área, devido à difícil tarefa de escolher a que melhor se enquadra para o seu objetivo final.

Existem, atualmente, dezenas de ferramentas e tecnologias para auxiliar na programação *front-end client-side* de aplicações e sítios web por forma a oferecer sempre mais e melhor dinamismo a estas aplicações. É uma área que tem crescido exponencialmente fazendo aumentar, na opinião da investigadora, a indecisão dos programadores e dos gestores de projeto quando se deparam perante tanta possibilidade de escolha para as suas aplicações, o que poderá ou não proporcionar-lhes muitos custos a nível de aprendizagem, tempo, entre outros.

Qual a *Framework model-view-controller* (mvc) javascript que mais se adequa à minha aplicação? Que custos terei ao escolher esta *Framework*? Irá afetar a performance do meu sítio web? Qual a popularidade desta *Framework*? Terei de despender muito tempo para aprender esta *Framework*? Estas e outras questões tornaram-se o ponto de partida para as empresas que estão a iniciar um novo projeto web. Se a escolha da ferramenta a ser utilizada no desenvolvimento do projeto não for a ideal poderá refletir-se em custos bastantes elevados. Por exemplo, uma empresa onde o seu projeto está mais direcionado para o *design* e visualização de produtos irá tomar uma decisão diferente de uma empresa que pretende interagir com o utilizador e permitir que ele use a aplicação com os dados da sua empresa para inserir produtos e atualizá-los constantemente. É importante, para os programadores e gestores de uma empresa estudarem e analisarem cuidadosamente as ferramentas disponíveis e selecionar aquela que melhor se adapta ao projeto.

Esta dissertação baseou-se numa investigação em contexto empresarial sobre a identificação de quais os critérios para a escolha de ferramentas e tecnologias *front-end*

client-side, mais especificamente *Frameworks* mvc de javascript, no desenvolvimento de *front-end client-side* de aplicações e sítios web. Foi uma investigação direcionada a uma aplicação de negócios que tem como público-alvo o mercado *business-to-business* (b2b), e como objetivo principal a criação de um centro de divulgação e gestão comercial que permitirá a interação entre empresas, pesquisa e identificação de produtos, assim como gestão de redes e cadeias de distribuição.

O estágio decorreu na empresa *start-up* Prodcent[1] que está localizada na incubadora de empresas da Universidade de Aveiro (IEUA) e tem aproximadamente um ano e meio de atividade. Esta é uma empresa que, dada a aceleração dos mercados mundiais procura ajudar empresas a acompanhar este progresso, permitindo alojar, tratar, promover e divulgar a informação empresarial e de produtos através da internet. Elimina ainda as barreiras geográficas, culturais, linguísticas e cronológicas com recurso às tecnologias simplificando assim o acesso e flexibilizando a utilização da aplicação.

Atualmente, esta empresa encontra-se a desenvolver um projeto denominado centroproduto[2] que se baseia na promoção internacional e na gestão da interação comercial entre empresas. Trata-se de um sítio web com capacidade multilíngue, onde as empresas poderão divulgar os seus produtos, serviços associados e gerir as suas interações comerciais.

B. Pergunta de investigação

A questão de investigação foi o foco central em toda a investigação. É ela que, segundo os autores Quivy e Campenhoudt (1995), traduz um projeto de investigação e só será útil se for corretamente formulada e pensada. É imprescindível que esta pergunta obedeça a determinados critérios, tais como, clareza, exequibilidade, univocidade e pertinência (Quivy & Campenhoudt, 1995). Deste modo, a questão que melhor traduziu este projeto de investigação e que respeitou todos os critérios enunciados foi a seguinte:

Quais os critérios de seleção de *Frameworks mvc* para o desenvolvimento de uma aplicação *client-side* web e *mobile* dirigida ao mercado b2b?

[1] http://www.prodcent.pt/ (último acesso 18-11-2013)
[2] www.centroproduto.com (último acesso 18-11-2013)

C. Finalidades e objetivos

Com esta investigação pretendeu-se identificar e validar os critérios seguidos para a seleção de uma *Framework* MVC (*Model-View-Controller*) *front-end client-side*.

Por forma a orientar melhor o estudo foram definidos alguns objetivos principais e outros mais específicos. Os objetivos principais passaram por identificar e caracterizar as diferentes ferramentas *front-end client-side,* assim como os critérios para a sua seleção. Essas mesmas ferramentas foram, posteriormente, analisadas de acordo com os diversos critérios de seleção para se tentar identificar a ferramenta que mais se adequava para a aplicação web desenvolvida em contexto empresarial.

Os objetivos específicos a atingir foram os seguintes:

- Identificar e caracterizar as *Frameworks front-end client-side* atualmente existentes;
- Analisar e selecionar os critérios para a escolha das tecnologias *front-end client-side*;
- Conhecer algumas opiniões de profissionais sobre as suas preferências de tecnologias e o porquê dessa decisão;
- Comparar as opiniões de profissionais com os critérios para a escolha das ferramentas *front-end client-side* identificados em contexto empresarial;
- Avaliar as tecnologias de acordo com os resultados obtidos no ponto anterior e segundo os critérios selecionados;
- Criar caminhos de decisão com todas as possibilidades e obstáculos para a escolha da *Framework* mais adequada;
- Escolher a tecnologia que mais se enquadra ao projeto desenvolvido em contexto empresarial e validá-la.

D. Estrutura da dissertação

A presente dissertação encontra-se dividida em duas partes, a primeira dedicada a conteúdo mais teórico e essencial para a investigação que decorreu na segunda parte onde incidirá todo o processo de escolha de uma *Framework* mvc de javascript. Inicialmente apresentar-se-á o enquadramento teórico, que pretende contextualizar e conduzir o desenvolvimento de toda a investigação. Incidirá sobre a conceitualização do termo B2B e prosseguirá com a definição e caracterização das principais e mais recentes *Frameworks* web, assim como dos vários critérios que condicionam e influenciam a seleção destas. Para terminar esta primeira parte, será apresentada a metodologia de investigação que melhor correspondeu aos objetivos da dissertação e pela qual incidiu todo o processo de investigação.

Na segunda parte deste documento será descrita a análise da recolha dos dados efetuados por vários gestores e programadores relativos às suas experiências e opiniões acerca do processo de escolha de uma *Framework* javascript, e da utilização dessa *Framework* nos seus projetos. Após esta análise será descrito o projeto em contexto empresarial. Uma vez que o processo de escolha da *Framework* javascript na entidade Prodcent, Lda decorreu antes da documentação da seleção dos principais critérios de escolha, considerou-se importante descrever todo o procedimento que conduziu à *Framework* final na empresa. Esta descrição incidirá numa exposição do projeto e da empresa, assim como, na explicação do processo de escolha da *Framework* javascript na empresa e da experiência e satisfação para com essa *Framework*.

Seguidamente serão mencionadas as *Frameworks* que foram consideradas mais importantes na análise da recolha dos dados e em contexto empresarial, e, de seguida será detalhada a avaliação elaborada para estas de acordo com as suas principais características e diferenças. Será então, após a descrição do processo anterior, explicado o procedimento de que resultou a tabela de pontuações entre as *Frameworks* e os critérios de escolha principais para, finalmente, se ter chegado à *Framework* ideal segundo esses critérios. Por último, será descrita a validação que foi realizada para a *Framework* escolhida segundo os requisitos do projeto em contexto empresarial, assim como de todas as restantes analisadas. A *Framework* final também foi validada segundo a opinião e experiência de alguns elementos da entidade empresarial, pelo que será visível uma descrição do feedback recebido.

Após todo este processo, prosseguir-se-á às conclusões finais dos resultados da investigação elaborada, às suas limitações e à descrição do trabalho que possa ser desenvolvido futuramente.

II. ENQUADRAMENTO TEÓRICO

Seguidamente serão relatados os aspetos mais teóricos que serviram de guia para o desenvolvimento da investigação. Estes tópicos definiram e caracterizaram os conceitos que foram mais relevantes para ter em consideração durante todo o processo da dissertação e serviram como base teórica e fundamentada para os passos posteriores.

A. Aplicações e *Framework*s web para *Business*-to-*Business* (B2B)

A Internet tem-se tornado parte integral na vida dos indivíduos (Engström & Salehi-Sangari, 2007). É atualmente, um importante canal de comunicação e comercialização de bens e serviços (Pinto, 2007). Alterou em larga escala, a maneira como as pessoas comunicam, trabalham e despendem o seu tempo (Engström & Salehi-Sangari, 2007). Uma das mudanças mais significantes desencadeada pela Internet é a forma como os negócios são atualmente conduzidos. A criação de mercados *online* veio reduzir custos para as empresas na forma como divulgam os seus produtos e serviços (Engström & Salehi-Sangari, 2007), criando uma plataforma universal para a compra de bens e serviços, ou para dirigir importantes processos de negócio (Pinto, 2007).

A Internet não modificou apenas a maneira como os negócios são conduzidos, mas também potenciou à criação de novos e diversificados. Este impacto é normalmente conhecido como comércio eletrónico (e-*commerce*), que ilustra o processo de compra, venda ou troca de serviços, produtos e informação através de redes computacionais, que inclui, como referido, a internet (Engström & Salehi-Sangari, 2007).

Os mercados eletrónicos *Business-to-Business* (B2B) são exemplos de novas práticas de negócios que emergiram para facilitar os processos de comércio eletrónico. Com este aumento de sofisticação e autonomia de comércio estas práticas tornam-se mais dinâmicas e eficientes e são bastante adotadas por empresas e organizações (Lau, 2007), abrindo portas e oportunidades para transações *online* mais eficientes entre estas (Engström & Salehi-Sangari, 2007).

1. B2B

O B2B é considerado como um tipo de comércio eletrónico para conduzir transações de negócios entre empresas (Pinto, 2007). Este conceito envolve a substituição dos processos físicos que contornam as transações comerciais por processos eletrónicos. Disponibiliza, ao fornecedor a criação e manutenção de catálogos de produto, receção e resposta a pedidos de propostas, receção de encomendas e manutenção de informação de *tracking* de encomendas (Pinto, 2007). Oferece, por outro lado, ao cliente, consulta de catálogos de produtos e serviços, possibilidade de colocar ordens de compra, acompanhamento de informação de *tracking* das suas encomendas e pedido e receção de propostas (Pinto, 2007).

O B2B é um conceito que assenta na partilha entre vendedores e clientes através de uma plataforma onde ambos podem efetuar transações de forma integrada e conciliável (Pinto, 2007). Inclui múltiplas plataformas de mercado na Internet que suportam vários modelos de negócio tais como leilões, trocas, *trading hubs*, centros de comercialização, etc.[3]. Fornece aos utilizadores bom conteúdo de informação sobre produtos e serviços, suporte de compra e venda, financiamento, privacidade e segurança, processo de pagamento, gestão das ordens de compra, definição do perfil ou personalização, configuração dos produtos, entre outros (Pinto, 2007).

Os autores Engström e Salehi-Sangari (2007) consideram que os principais benefícios deste comércio são:

- Baixos custos de pesquisa para compradores;
- Redução de níveis e custos de inventários;
- Menos custos transacionais e redução de custos administrativos ao eliminar a utilização de documentos em papel e automatizando partes do processo de aquisição;
- Potencia o aumento da cooperação entre vendedores e compradores o que faz aumentar a qualidade dos produtos;
- Aumenta as oportunidades para colaboração entre fornecedores e distribuidores;
- Habilita a visualização dos preços atuais de venda e compra num mercado.

No B2B são envolvidas três grandes áreas, que são *e-marketplaces*, *e-procurement* e *e-distribution* (Pinto, 2007). As e-*marketplaces* são plataformas eletrónicas

[3] Alguns exemplos de modelos de negócio: http://www.superbid.net/ (último acesso: 20/01/2014), http://www.eve-search.com/thread/966663-0/page/1 (último acesso: 20/01/2014)

utilizadas pelas empresas para estabelecer laços empresariais entre si. Estas permitem transações e comunicações comerciais *online* (Pinto, 2007). O *e-procurement* é uma plataforma eletrónica desenvolvida para suportar "*o aprovisionamento das organizações, permitindo que estas otimizem a cadeia de fornecimento em termos de tempo e de custos, através da automatização das interações com as centrais de compras dos seus fornecedores*" (Pinto, 2007, p.45). Por último, a *e-distribution* designa plataformas eletrónicas concebidas para integrar as empresas com os seus distribuidores, filiais e representantes. Através destas plataformas é possível efetuar consultas de catálogos eletrónicos, emitir de faturas, entre outros (Pinto, 2007).

De forma a aprofundar a análise deste conceito B2B, foi elaborada uma pesquisa para se conhecer algumas das principais aplicações web desta vertente que estão atualmente em funcionamento, fazendo aumentar as vantagens desta conceção.

A **Alibaba**[4]**,** que foi lançada em 1999, é atualmente a plataforma líder global de comércio eletrónico para pequenos negociantes. Objetiva ser uma plataforma em língua inglês corrente, para ajudar negociantes pelos mercados mundiais. Atualmente esta aplicação serve milhões de compradores e fornecedores por mais de 240 países e regiões.

A **Tradecom**[5] nasceu em 2000 pelas empresas PT prime e a Commerce One e é a empresa líder no fornecimento de serviços de comércio eletrónico B2B no mercado nacional. É responsável pela criação do primeiro portal, dedicado ao comércio eletrónico entre empresas no mercado português. Tem como funcionalidades, a compra e venda de produtos, leilões, serviços EDI, e faturas eletrónicas.

A **EC21**[6] iniciou atividade em 1997 e é um dos maiores mercados mundiais B2B *online.* Desde o seu início que tem facilitado transações globais para pequenas e médias empresas. Esta plataforma tem cerca de um milhão de membros, três milhões de produtos, um milhão de compradores e 3,5 milhões de visitantes por mês e ainda está em crescimento.

A **Made-in-China**[7] foi criada pela empresa Focus Technology, uma empresa pioneira e líder em comércio eletrónico na China. Fundada em 1996 com a missão de permitir aos clientes divulgar os seus produtos na web. É uma plataforma destinada a pequenas e médias empresas, ajudando estas a competir com transações internacionais, introduzi-las em novos mercados e a conhecerem as necessidades dos seus clientes.

[4] www.alibaba.com – (último acesso: 20/12/2013)
[5] http://www.tradecom.pt/ - (último acesso: 20/12/2013)
[6] www.ec21.com – (último acesso: 20/12/2013)
[7] http://www.made-in-china.com/ - (último acesso: 20/12/2013)

A **Globial**[8] foi fundada oficialmente em 2010 na Califórnia, em Sunnyvale. É uma rede de descoberta para compradores e vendedores e tem como missão principal tornar transações globais mais fáceis, conectando negociantes de confiança por todo o Mundo.

Entre estas plataformas mencionadas anteriormente, ainda existem outras com o mesmo conceito e objetivo, tais como **iwaytrade**[9], **bizdirect**[10], **busytrade**[11], entre outras.

2. Aplicações web Front-end (cliente) Client-side

O conceito de *client-side* começou a tornar-se mais comum apenas há poucos anos atrás (Nicollet, 2012). Abordagens anteriores recorriam ao servidor para gerar todo o código *Hypertext Markup Language* (HTML) de uma página para enviar para o cliente(Nicollet, 2012). Esta estratégia tinha os benefícios de evitar a execução de código do lado do cliente, o que, em inícios do ano 2000 era bastante penoso em termos de performance e necessidade de escrever o código em Javascript, considerado inútil nessa data (Nicollet, 2012).

Anos depois, o Javascript e a sua reputação evoluíram e a geração de HTML no *browser* deixou de ser considerado uma má estratégia, como a grande maioria dos programadores suponha. Apesar de o servidor ainda gerar HTML suficiente, a programação elaborada no lado do cliente acabou por evoluir, adicionando mais funcionalidades e características à interface do utilizador, tais como: efeitos de transição, *date pickers,* o aparecimento do Ajax[12], que aumentou a importância de aplicações responsivas e rápidas sem necessidade de estarem sempre a aceder ao servidor e a recarregar a página por cada interação do utilizador, tornando-se, deste modo, um marco importante na história do javascript, entre outros. Esta nova mudança de paradigma foi tão eficiente que em 2012, quase todos os sítios web ou aplicações envolviam processamento *client-side*, contudo, maior parte do código HTML continuava a ser gerado no servidor (Nicollet, 2012).

> *"The balance is undeniably moving from traditional server-side templating to JavaScript templating"* - (Hales, 2012, p.3)

[8] http://www.globial.com/ - (20/12/2013)
[9] http://www.iwaytrade.pt - (20/12/2013)
[10] http://www.bizdirect.pt - (20/12/2013)
[11] http://www.busytrade.com - (20/12/2013)
[12] "*AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page*." - http://www.w3schools.com/ajax/ (Último acesso: 12/12/2013)

O autor Addy Osmani (s.d) acrescenta ainda que o positivo impacto de compromisso dos utilizadores com sítios web e aplicações deve-se especialmente a melhorias na latência e performance da aplicação. A probabilidade de uma alta latência ocorre muitas vezes em aplicações web centradas no servidor, que requerem a atualização de uma página inteira para prosseguir para outra. Mesmo em cache, o *browser* terá de carregar os ficheiros de *Cascading Style Sheets* (CSS), Javascript e HTML para renderizar a interface da nova página no ecrã. Aplicações baseadas no servidor afetam também a capacidade de resposta ao utilizador, uma vez que este terá de esperar por um sinal do servidor por cada passo que efetuar na página envolvendo comunicação com os dados (Addy Osmani, s.d.).

O *client-side* tem vindo a ganhar mais consideração (Nicollet, 2012). Os *browsers* estão a tornar-se plataformas adicionais para as aplicações web (Hales, 2012), têm ganho muito mais reputação relativamente ao *server-side* no momento de escolha do que uma nova aplicação deverá suportar. Abordagens como gerar o código HTML no *browser* e ter um servidor a fornecer os dados através de *Application Programming Interfaces* (APIs) em formatos computacionais amigáveis tais como, *Javascript Object Notation* (JSON) ou *eXtensible Markup Language* (XML), têm sido cada vez mais comuns e potenciais (Nicollet, 2012).

Segundo o autor Runeberg (2013) a comunicação entre *front-end* e *back-end* (servidor) implica os seguintes passos cíclicos:

1- Utilizador escreve ou clica numa determinada hiperligação no seu *browser;*
2- O *browser* faz um pedido *Hypertext Transfer Protocol* (HTTP) ao servidor, contendo corpo e cabeçalho;
3- O servidor processa o pedido de acordo com o tipo de consulta e os seus parâmetros;
4- O servidor manipula ou cria dados na base de dados, em caso de páginas dinâmicas;
5- O servidor envia uma resposta HTTP com um cabeçalho de estado 200 (OK), e um corpo, por exemplo, o modelo que foi alterado, contendo os dados numa forma compreensível para o cliente, como HTML ou Json;
6- O *browser* recebe a resposta para dar ao utilizador;
7- A resposta é renderizada pelo *browser* para ser disponibilizada ao utilizador;
8- Em alguns casos a resposta desencadeia novos e mais pedidos HTTP ao servidor.

Novas interações do utilizador, como cliques, envio de formulários, entre outros irão desencadear novos ciclos de comunicação entre cliente e servidor.

Uma das novas tendências que têm surgido nos últimos tempos para melhorar os problemas de performance, nas páginas web, passa também pela construção de S*ingle Page Applications*[13] (SPAs), aplicações que depois do carregamento inicial da página são capazes de comportar posteriores navegações e pedidos de dados sem necessidade de atualizar completamente o DOM (Addy Osmani, s.d.).

Ao se construir uma SPA recorrendo ao uso da linguagem Javascript, se esta envolver uma interface muito complexa ou simplesmente se estiver a tentar reduzir o número de pedidos HTTP exigidos pelas *views*, o utilizador irá muitas vezes tentar desenvolver partes lógicas que já existem numa *Framework* MV*, sem se aperceber desse facto (Addy Osmani, s.d.).

3. *Framework*s web

Atualmente criar uma aplicação web interativa de raíz não é tarefa fácil. A necessidade de articular e integrar diferentes tecnologias e linguagens, e o leque alargados de *Frameworks* disponíveis tornaram-se uns dos maiores problemas no desenvolvimento de aplicações Web (Ousterhout, 2009).

O acesso à web não está limitado apenas aos computadores (Allen, Graupera, & Lundrigan, 2010), tem-se verificado um crescimento no uso de dispositivos móveis como dispositivos de acesso à web. Contudo, a utilização de diferentes dispositivos móveis no acesso à web veio trazer dificuldades de compatibilidades com os diferentes sistemas operativos, tamanhos de ecrãs, entre outros, o que potenciou o aparecimento de uma grande variedade de *Frameworks*, assim como o aparecimento do HTML5 e CSS3, tornando as aplicações mais responsivas e adaptáveis (Miller, 2011).

O conceito de *Framework* foi desenhado para suportar o desenvolvimento de sítios web, aplicações e serviços por forma a tornar mais fácil trabalhar com tecnologias complexas, assim como estruturar o código de maneira mais organizada e consistente. Existe, neste tipo de tecnologia, uma separação de conceitos onde diferentes aspetos de uma aplicação são mantidos separadamente em subsistemas com funcionalidades distintas (Grüneberger, 2012). Uma *Framework* auxilia a equipa de desenvolvimento a implementar código de uma maneira que promove programar de forma metódica, com menos bugs e a criar aplicações mais flexíveis (Clifton, 2003). Dentro desta

[13] *"SPAs are web applications that load into the browser and then react to data changes on the client side without requiring complete page refreshes from the server*" - (Addy Osmani, 2012, p.11)

modularidade, as *Framework*s suportam o rápido desenvolvimento de aplicações e promovem o seu reuso e manutenção (Grüneberger, 2012).

É importante para o programador reconhecer quando irá necessitar de uma *Framework*. Se este estiver a desenvolver uma aplicação onde o processamento da *view* para a renderização e manipulação dos dados é executada no *browser*, utilizar uma *Framework* será o mais apropriado, permitindo uma redução no tempo necessário para a inventariação das partes lógicas da mesma. Quando de acede a este tipo de aplicações é transferido de uma só vez, para o *browser,* todos os scripts, estilos e HTML que o utilizador irá precisar para as tarefas comuns. No entanto, se o utilizador estiver a construir uma aplicação onde predomina no servidor (*server-side*) a maior parte do processamento da renderização da página e este apenas estiver a usar um pequeno excerto de código Javascript ou Jquery, por forma a tornar a página mais atrativa, uma *Framework* MV* talvez não seja a solução mais apropriada (Addy Osmani, 2012b).

O surgimento de *Frameworks* web veio atenuar o conceito de *spaghetti code:*

> *"Spaghetti code is a pejorative term for source code that has a complex and tangled control structure, especially one using many gotos, exceptions, threads, global variables, or other "unstructured" constructs. It is named such because program flow tends to look like a bowl of spaghetti – twisted and tangled. It is also used in pejorative sense to imply that a given piece of work is difficult to understand*" -(Mikkonen & Taivalsaari, 2007, p.6-7)

Este conceito de *spaghetti code,* na opinião da investigadora[14], tem vindo a preocupar muitos programadores*,* principalmente os mais inexperientes que não detêm de tanto conhecimento programático, nem sabem da abrangência de ferramentas disponíveis atualmente para facilitar a estruturação do código e a manutenção de aplicações mais complexas, fazendo com que a certa altura do desenvolvimento da sua aplicação a confusão e a desorganização se instalem. Também, para aplicações que necessitem de atualizações constantes, a reutilização do código torna-se ilegível e de difícil restruturação.

O foco deste documento são as *Frameworks* mvc de javascript, pelo que, nos próximos parágrafos, serão detalhadas algumas *Frameworks* mvc de javascript e o próprio conceito de *model-view-controller*.

---

[14] Foi referida a opinião da investigadora neste parágrafo por falta de artigos científicos que a fundamentassem, e deste modo se justifica a ausência de referências.

a) Javascript MV* e MVC

O Javascript tornou-se uma das maiores e mais populares linguagens de programação da internet (Grüneberger, 2012). Inicialmente, foi concebido apenas para a interface do utilizador contudo, mais tarde, com o aparecimento do Ajax, melhorou a interação do utilizador ao impedir o constante carregamento de páginas. Este crescimento no uso do Javascript em aplicações mais modernas veio obrigar os programadores a programar de forma mais sustentável e organizada, com separação de conceitos e testes improvisados (Anil, 2013).

Cada nova aplicação implica ser estruturada segundo certos padrões de design[15] que objetivam uma clara separação das suas partes lógicas, de modulação e de visualização, normalmente definidas da seguinte forma (Grüneberger, 2012):

- A interface;
- Os dados da aplicação;
- A interação.

Este padrão vem caracterizar o conceito de *model-view-controller* (MVC) o qual tem como objetivo principal separar o modelo de dados da sua lógica e da sua visualização real (Anil, 2013).

O *Model-View-Controller* é um conceito que surgiu com Trygve Reenskaug em 1979 (Grüneberger, 2012). É um termo focado na inspeção e manipulação dos dados visualizados por múltiplas *views* e descreve a proteção dos dados com métodos relacionais num modelo de dados isolado da sua apresentação e manipulação pelo utilizador. (Grüneberger, 2012)

Como se pode visualizar na Ilustração 1 este modelo separa as funcionalidades de uma aplicação em três partes lógicas que são caracterizadas da seguinte forma (Addy Osmani, 2012; Grüneberger, 2012):

- *Model*: representa a coleção de dados associados a métodos de manipulação. Delimitam qual o tipo de dados que irão ser trabalhados, por exemplo, uma foto, um vídeo, entre outros. Estes devem notificar as *views* do seu estado atual e estas só acedem a eles através de uma interface, logo, um modelo pode esconder como é que os dados são colocados nessa interface, se usados em memória, em ficheiros locais ou numa base de dados relacional.
- *View:* considerada como uma interface do utilizador, determina como o modelo é mostrado no ecrã, mas não comunica diretamente com ele. Um

[15] "*A pattern is a reusable solution that can be applied to commonly occurring problems in software design - in our case - in writing JavaScript web applications. Another way of looking at patterns are as templates for how we solve problems - ones which can be used in quite a few different situations*" - (Addy Osmani, 2012)

model pode ser representado por várias *views,* que para se manterem sincronizadas com o mesmo, subscrevem a modificações do modelo e requerem atualizações sempre que alguma alteração tenha ocorrido. Uma view também é responsável por encarregar eventos do utilizador a um *controller e* por controlar a saída dos dados.

- *Controller:* responsável pela manipulação dos eventos de entrada (cliques, escrita) do utilizador. Trata a lógica de uma aplicação e tem como finalidade atualizar o modelo através das interações do utilizador com a *view*. Por exemplo: quando um utilizador edita a legenda de uma foto, não se está diretamente a comunicar com a *view,* mas sim a atualizar o *model* através da interação do utilizador com a mesma.

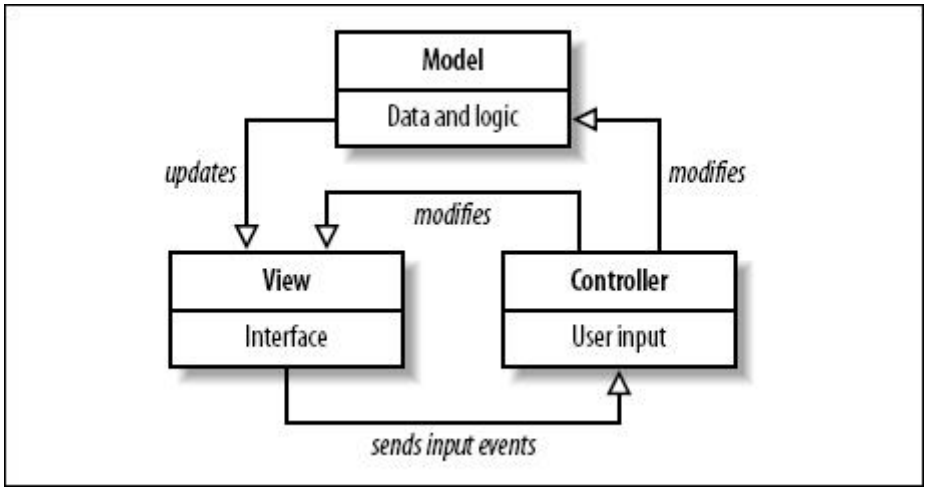


Ilustração 1- Model-View-Controller

Entre todas as *Frameworks* existentes na atualidade, algumas tentam adotar o modelo MVC, contudo podem não seguir este padrão de forma rigorosa (Anil, 2013). Algumas *Frameworks* misturam a responsabilidade do *Controller* na *View*, enquanto outras adicionam componentes extras dentro dessa mistura. Por esta razão, estas *Frameworks* são caracterizadas como seguidoras do modelo MV*, que significa ter um *Model* e uma *View,* mas um *Controller* distinto que possa não estar presente ou existirem outras componentes em vez deste - Model-View-Presenter (MVP) *e* Model-View-ViewModel (MVVM) (Addy Osmani, s.d.).

(1) *Framework*s Javascript MV*

Seguidamente serão apresentadas algumas das *Frameworks* mais populares de Javascript, sendo que, algumas são mesmo consideradas Bibliotecas em vez de *Frameworks*, porventura, serão mencionadas conjuntamente com as outras devido ao seu grau de relevância. Para clarificar as diferenças entre *Framework* e Biblioteca considera-se uma Biblioteca o facto de disponibilizar uma estrutura de código que se adapta à arquitetura de código do programador já existente, e permite que sejam adicionadas funcionalidades específicas à arquitetura existente (Anil, 2013). As *Frameworks* oferecem uma arquitetura própria e o programador terá de seguir as regras que estas dispõem (Anil, 2013).

O processo de seleção das *Frameworks*, apresentadas de seguida, baseou-se na lista de *Frameworks* e bibliotecas disponíveis no sítio web TodoMVC[16] e nas *Frameworks* mencionadas em artigos recentes de programadores profissionais.

Como exemplos de Bibliotecas temos a:

BackboneJS[17] - é uma biblioteca de javascript bastante simples e com soluções minimalistas por forma a separar a estrutura lógica de uma aplicação *client-side*. Apesar de ser uma biblioteca recente é bastante popular e madura, o que faz com que tenha uma comunidade ativa e alargada (Addy Osmani, s.d.). É muito útil para criar aplicações do tipo *single page* e do tipo *REST*, oferecendo toda uma parte funcional dedicada a esse aspeto. É conhecida também pela sua flexibilidade por forma a criar a melhor experiência aos programadores no desenvolvimento de aplicações web (Addy Osmani, s.d.). Os programadores ao programarem com esta biblioteca dispõem de tantas funcionalidades quantas quiserem para os seus projetos, sem necessidade de terem de criar soluções insustentáveis para se manterem dentro dos padrões que a biblioteca espera que os programadores usem (Marriner, s.d.).

Algumas características (Addy Osmani, s.d.; Runeberg, 2013):

- Estrutura de código muito básica e fácil deixando o resto para o programador programar. Por ser bastante minimalista, oferece liberdade ao programador, sem este ter necessidade de seguir determinados padrões da biblioteca. Este aspeto pode ser considerado tanto positivo como negativo, oferecendo pouca estrutura, também mais facilmente se pode perder toda a organização do código;

[16] www.todomvc.com (último acesso: 12/12/2013)
[17] http://backbonejs.org/ (último acesso: 12/12/2013)

- É rica em extensões e *plugins,* pelo facto de ser tão simples, oferece todo um conjunto de bibliotecas para poderem ser adaptadas à estrutura de código original;
- Não força a utilização de modelos de HTML, mantem este simples e limpo, sem *binds* e qualquer lógica;
- É facilmente extensível para outras aplicações e vice-versa;
- Utilizada por websites de grande prestígio, tais como Linkedin.*com, Soundclound,* etc…;
- Bastante leve e compacta, sendo possivelmente a mais leve de todas a *Frameworks* mencionadas neste documento;

Alguns inconvenientes (Addy Osmani, s.d.; Runeberg, 2013):

- Dado seu peso, necessita de algumas dependências extra para poder trabalhar corretamente, tais como *jquery ou zepto*, e um modelo de HTML, tal como *underscore.js*;
- Documentação um pouco confusa e incompreensível;
- Não utiliza *two way data binding* (por defeito não utiliza, mas fornece uma extensão que disponibiliza esta funcionalidade para quem desejar);
- Pouca interatividade entre o design e programação, sendo grande parte dos seus templates programados via javascript;

***KnockoutJS*[18]-** É uma Biblioteca de Javascript que pode ser incluída num sítio web ou aplicação, para adicionar funcionalidades ao Javascript, aumentando assim a experiência do utilizador (Barnard, 2012). Baseia-se no modelo MVVM e é considerada a ferramenta mais simples de todas, assim como, uma das mais pequenas, e por estes motivos, talvez não seja capaz de resolver todos os problemas em aplicações que aparentem ser mais complexas. É uma Biblioteca "*low risk*" e flexível por ser bastante focada e compatível com muitas outras tecnologias. Não foi criada para ser usada isoladamente ou para substituir *Frameworks* existentes, mas sim para ser utilizada em conjunto com elas. O programador poderá utilizá-la em apenas uma pequena parte da sua aplicação, ou na aplicação como um todo (Trager & Kagan, 2013). É uma *Biblioteca* e por ser usada facilmente com outras *Frameworks*, trabalha bem com jQuery e vem incorporada com um modelo de HTML de jQuery por defeito, e por este motivo é dependente de jQuery para poder funcionar de forma completa (landland, 2013).

[18] http://knockoutjs.com/ (último acesso: 13/12/2013)

Knockout é uma Biblioteca bem documentada, embora a sua sintaxe de *binds* possa ser um pouco escassa a nível de *views* sólidas e hierarquizadas, por forma a facilitar a reutilização das suas componentes. Não tem um sistema de *routing*, o que poderá ser uma grande desvantagem da mesma, contudo ganha pontos positivos pela sua compatibilidade com *browsers* mais nativos.

***CanJS*[19] –** "*CanJS is a JavaScript library that makes developing complex applications simple and fast. Easy-to-learn, small, and unassuming of your application structure, but with modern features like custom tags and 2-way binding. Creating apps is easy and maintainable*."[24]

CanJs é uma biblioteca baseada no modelo MVC que oferece as ferramentas necessárias para desenvolver aplicações com Javascript. Proporciona Templates com ligações reais, suporte de roteamento e é ótimo na gestão da memória, impedindo vazamentos[20]. Suporta Jquery, Zepto, Mootools, YUI, Dojo e possui um abundante conjunto de extensões e Plug-ins (Cummings, 2012).

Como exemplos de *Frameworks* temos a:

***AngularJS*[21]**- é uma *Framework* ideal para desenvolver aplicações web *single page*, modernas e do estilo Ajax. O principal estilo de aplicações a que se aplica são as do tipo CRUD e utiliza a linguagem HTML como linguagem de modelos (Kozlowski & Darwin, 2013). AngularJs é uma extensão do DOM, adicionando-lhe novos parâmetros e interagindo de forma dinâmica com vários elementos. Com esta *Framework* é possível acrescentar novos atributos de HTML para conseguir adicionar funcionalidades extras, sem necessidade de programá-las em javascript. Trata-se de uma linguagem declarativa que usa novos parâmetros na linguagem HTML para alterar o seu comportamento padrão (Schimtz & Lira, 2014).

Esta *Framework* não requer o recarregamento do DOM, é capaz de captar ações do utilizador, eventos do *browser* e mudanças no *model*, por forma a ter conhecimento de quando e quais os modelos atualizar (Kozlowski & Darwin, 2013).

Algumas características ("10 Reasons Web Developers Should Learn AngularJS," 2013; Runeberg, 2013):

- Tamanho de 77kb;
- Não requer dependências de outras bibliotecas ou *Frameworks*;
- *Two way data binding;*

[19] http://canjs.com/ (último acesso: 12/12/2013)
[20] http://bitovi.com/blog/2012/04/zombie-apocolypse.HTML (último acesso: 12/12/2013)
[21] http://angularjs.org/ (último acesso: 12/12/2013)

- Muito bem documentado e com exemplos explícitos e uteis para um programador;
- Bastante acessível a nível de testabilidade. Tem ferramentas como *Karma*[22], *PhantomJs*[23], entre outras para os seus testes unitários;
- Validação de formulários;
- *Dependency Injection;*
- Permite boa interação entre o design e a programação;
- Oferece tecnologias para se inspecionar a performance, as dependências, todas a variáveis a serem processadas e as que afetam mais a performance, entre outros, tais como *Angularjs Batarang*.

Alguns inconvenientes:

- Faz *"dirty checking"* que é a comparação que se faz de um determinado valor no *model* com o valor anterior, e quando estes diferem é desencadeada uma ação para mudança desse valor. Esta comparação é realizada de forma repetitiva por forma a todas as atualizações se desencadearem em tempo real. Apesar de este método ser vantajoso para o conceito de *two way data binding* da *Framework* Angularjs, pode resultar também em problemas de performance[24] e lentidão nas interações com o *website* (Runeberg, 2013);
- A curva de aprendizagem desta *Framework* pode ser considerada difícil, uma vez que é necessário um bom entendimento do DOM, de diretivas, de filtros e de muitos aspetos que são novos para programadores de *front-end*.

***EmberJS*[25] -** esta *Framework* nasceu de uma equipa que trabalhava num projeto da Apple, denominado de *SproutCore* que objetivava trazer *software* do *desktop* a desenvolver padrões para a web (Marriner, s.d.). Trata-se de uma *Framework* front-end seguidora do modelo MVC que corre num *browser* e é destinada a programadores que pretendam desenvolver aplicações mais ambiciosas e complexas, podendo estas, competir com aplicações nativas (Bodmer, 2013).

Esta *Framework* espera que o programador construa grande parte do seu código em volta das suas metodologias e arquitetura. *EmberJS* não permite incorporar a sua estrutura em aplicações já existentes, como é o exemplo da *Framework AngularJS* ou

---

[22] "*On the AngularJS team, we rely on testing and we always seek better tools to make our life easier. That's why we created Karma - a test runner that fits all our needs.*" - http://karma-runner.github.io/0.8/index.HTML (10-08-2014)
[23] http://phantomjs.org/ (10-08-2014)
[24] http://stackoverflow.com/questions/9682092/databinding-in-angularjs/9693933#9693933 – (10/08/2014)

Knockout, *EmberJS* foi planeado com a sua própria arquitetura, de modo a poder criar sólidas aplicações para o programador (Bodmer, 2013).

EmberJS é uma *Framework* para comportar aplicações com interações mais complexas. Pode-se tornar, também, uma ferramenta complicada enquanto o programador não entender e dominar os seus principais conceitos (Bodmer, 2013).

***ExtJS*[26] –** é uma *Framework* moderna de Javascript criada pela Sencha. *ExtJS* fornece as ferramentas necessárias para construir robustas aplicações de desktop[15]. Inicialmente, esta *Framework* foi desenvolvida para servir de extensão à *Framework* do Yahoo *(*YUI) (Teixeira, s.d.). É uma *Framework* de rápida prototipagem e implementação, com um vasto número de *widgets* configuráveis e personalizáveis. É orientada a objetos e torna o trabalho do programador simples por todos os *browsers. ExtJs* é uma *Framework* que torna as funcionalidades fáceis de alcançar, contudo a sua curva de aprendizagem pode tornar-se longa e por isso, esta ser uma *Framework* mais aconselhável a programadores mais experientes.

***BatmanJS*[27] -** Criada por Shopify, *BatmanJS* é uma *Framework* similar a *AngularJs* e *KnockoutJs*. Tem um sistema de ligações de interfaces de utilizador baseado em atributos de HTML e é a única *Framework* escrita numa pequena linguagem de programação que compila no topo do Javascript, denominada *coffeescript*[28]. *BatmanJs* é também integrada com *NodeJs*[29] (Hales, 2012).

***Javascript MVC*[30]** - é uma *Framework open source* de Javascript baseada no modelo MVC e construída no topo da Biblioteca de *Jquery*.

É a *Framework client-side* agnóstica do *back-end* que pode ser usada com qualquer solução *back-end*, tal como *NodeJS*, *Ruby*, *Django*, etc.. Foi lançada em Maio de 2008 e contém todas as componentes necessárias para construir, gerir, documentar e testar projetos baseados em Javascript (Bednarski, 2013).

A sua curva de aprendizagem é bastante baixa especialmente se o programador já for familiar com uma outra *Framework*. Uma das vantagens desta *Framework* é que prevê *memory leaks* o que é bastante importante para aplicações *client-side* (Bednarski, 2013).

---

[25] http://emberjs.com/ (último acesso: 13/12/2013)
[26] http://www.sencha.com/products/extjs/(último acesso: 12/12/2013)
[27] http://batmanjs.org/ (último acesso: 12/12/2013)
[28] http://coffeescript.org/ (ultimo acesso: 12/12/2013)
[29] http://nodejs.org/ (ultimo acesso: 12/12/2013)
[30] http://javascriptmvc.com/ (ultimo acesso: 12/12/2013)

Desenvolver aplicações *single-page* com recurso a Javascript tem-se tornado cada vez mais popular na atualidade (Anil, 2013). Bons exemplos a retratar este facto são o *Gmail* e o *Google docs*. Este tipo de aplicações, como referido anteriormente, transfere de uma só vez todos os *scripts,* estilos e HTML que o utilizador irá necessitar para as tarefas comuns. É importante trocar entre ler uma mensagem de correio eletrónico ou um documento e escrever um sem enviar novamente um pedido de uma nova página para o servidor (Addy Osmani, s.d.).

Um outro caso prático é refletido na aplicação mobile do *Linkedin:*

> *"In addition to* HTML*5, the team also used a handful of lesser-known free and open-source tools, such as* Backbone *and* Underscore*, in developing the apps. "The way our mobile web app works is it's all rendered on the browser side. The value of that is you send less data back and forth, so it's much faster." Also, the app is insanely lightweight. "If you take our entire app and you combine all the Framework pieces and zip it, it's under 50K," said Prasad." - (O'Dell, 2011)*[31]

BackboneJS é usado também por sítios web tais como *Disqus*, *Walmart* e *SoundCloud*.

Aplicações web como *Groupon ou Yahoo* adotaram a *Framework* EmberJS. Por outro lado, Youtube para PS3, *eTuneBook, doubleclick,* entre outros, preferiram a *Framework* Angularjs.

JavascripMVC é utilizado por sítios web como a IBM, enquanto que *EmberJS* é conhecido em aplicações como *Square.*

[31] *Exclusive: How LinkedIn used Node.js and* HTML*5 to build a better, faster app*. Disponível em: http://venturebeat.com/2011/08/16/linkedin-node/ (último acesso: 20/12/2013)

b) *Framework*s móveis

Cada vez mais, os consumidores *online* esperam melhor e mais versáteis aplicações em menos tempo. É um grande desafio desenvolver aplicações móveis de alta performance neste mercado cada vez mais competitivo (Singh & Palmieri, s.d.) e dividido em múltiplas plataformas, tais como, Android, IOS, Blackberry, Windows Phone, etc.. Cada plataforma é diferente, e por isso, desenvolver uma aplicação nativa significa ter de aprender várias linguagens de programação e manter um código base para cada uma como mostra a Ilustração 2 (Miller, 2011).

Ilustração 2 - Dispositivos e sistemas operativos móveis

Aplicações móveis podem ser intimidantes para quem as tem de desenvolver (Miller, 2011). Ter de aprender múltiplas linguagens de programação, *softwares* de desenvolvimento, ou gerir quantidades consideráveis de código leva a um trabalho extra que aumenta o tempo de desenvolvimento, assim como a probabilidade de surgimento de erros, o que conduzirá a custos iniciais para o projeto do programador (Miller, 2011).

Embora as aplicações nativas dominem as estratégias móveis, recentes investimentos em aplicações baseadas na web têm vindo a oferecer estratégias para desenvolver aplicações pelas diferentes plataformas usando um código base universal (Miller, 2011). Estas ferramentas de desenvolvimento web multiplataforma oferecem código que pode ser executado em todos os sistemas operativos suportados, e têm vindo a ganhar bastante significado (Singh & Palmieri, s.d.).

Contudo, talvez a mais importante evolução seja o conceito de hibrido, que predomina entre aplicações nativas e baseadas em web, oferecendo plataformas que trabalhem com múltiplos sistemas operativos e gerações seguintes de dispositivos móveis. Com estas alternativas hibridas e web *"breaking into the mobile application universe is no longer as hard as it once was"* (Miller, 2011 p.27).

As aplicações baseadas em sítios web que se assemelhem a aplicações móveis nativas, normalmente recorrem a *Framework*s móveis de Javascript como *jQtouch*, *Jquery Mobile* ou *SenchaTouch* para replicar uma interface de utilizador móvel (Miller, 2011). Estas *Framework*s que são destinadas ao desenvolvimento móvel, acomodam dispositivos de toque, utilizam um sítio web para disponibilizar uma experiência de utilizador móvel completa e com animações, possuem também capacidades *offline* e acesso limitados às componentes nativas dos dispositivos. São essencialmente multiplataformas. As desvantagens destas aplicações são o acesso limitado aos recursos dos dispositivos e o facto de estas não poderem ser candidatas a lojas de vendas, como *AppStore*, o que afeta o seu rendimento de produção (Miller, 2011).

As aplicações hibridas oferecem aos programadores grande controlo sobre o *design* da sua aplicação, assim como acesso aos recursos do dispositivo. Estes benefícios tornam possível desenvolver aplicações multiplataformas através de um único código base e mesmo assim obter performance e disponibilidade comparáveis com aplicações nativas (Miller, 2011).

Seguidamente irão ser apresentadas alguns exemplos de *Framework*s e plataformas móveis que permitem melhorar as experiências dos utilizadores. Estas foram selecionadas segundo a sua popularidade e frequência ao longo dos artigos pesquisados:

- ***PhoneGap*[32] -** é um projeto patrocinado pela *Nitobi*[33] e lançado em 2008. *PhoneGap* é uma *Framework open source* para criar e desenvolver aplicações móveis nativas com recurso a HTML, CSS e Javascript para a grande parte dos dispositivos (Allen et al., 2010). É essencial para transformar aplicações móveis baseadas em web em aplicações nativas. Torna-se facilmente usada por web *developers*, contudo estes terão de deter de algum conhecimento em como construir aplicações com recurso a um ou mais dispositivos e ferramentas *Software Development Kits (SDKs)*, no entanto, todo o código da ferramenta pode ser implementado em HTML, CSS e Javascript, aliás, um programador com bons conhecimentos de Javascript pode obter grandes vantagens desta plataforma.
  Esta *Framework* proporciona ricas coleções de APIs de Javascript com métodos para alojar aplicações móveis web dentro de aplicações nativas móveis (Allen et al., 2010).

[32] http://phonegap.com (último acesso: 20/12/2013)
[33] http://nitobi.com (último acesso: 20/12/2013)

A grande vantagem de criar ferramentas nativas com esta *Framework* é que o *developer* consegue desenvolver uma aplicação web móvel no topo de uma nativa, a qual um utilizador final irá, posteriormente, instalar. Como aplicação nativa, poderá aceder a certas capacidades dos dispositivos, que não são, todavia, acessíveis para as ferramentas web (Allen et al., 2010).

Para criar aplicações com esta ferramenta, o utilizador irá começar por desenvolver uma aplicação móvel web, onde poderá utilizar qualquer outra *Framework* ou ferramenta que se sinta mais confortável, uma vez que esta *Framework* proporciona muito pouca estrutura baseada em padrões de *design* e MVC. *PhoneGap* é apropriada para qualquer funcionalidade que se pretenda desenvolver em aplicações web *mobile*, contudo, como todas as outras *Framework*s, não se torna aconselhável para aplicações que requerem intensos cálculos matemáticos ou animações em 3D (Allen et al., 2010). Em 2012 esta Framework renomeou o seu nome para Apache Cordova[34].

- ***Rhodes*[35] -** é uma *Framework* multiplataforma para dispositivos móveis desenvolvida pela *Rhomobile*. Foi lançada pela primeira vez em Dezembro de 2008 e está disponível para a maior parte de dispositivos móveis (*iPhone*, *Android*, *Blackberry*, *Windows Mobile* e *Symbian*). Uma das vantagens desta *Framework* predomina na habilidade desta em conduzir uma empresa na construção e manutenção de um único código base, atravessando a grande variedade de sistemas operativos existentes atualmente (Allen et al., 2010). *Rhodes* permite ao utilizador criar aplicações móveis usando HTML, CSS, Javascript e *Ruby*[36] como linguagens de programação e, desta forma, influência a experiência do *developer* no desenvolvimento web para criar aplicações móveis nativas.

  Esta *Framework* simplifica o desenvolvimento de aplicações que apresentem vários ecrãs que incluem interfaces *standards* de *Widgets*, e também funcionalidades comuns do dispositivo. Não é aconselhável para a conceção de jogos de ação ou para outras interfaces que exijam gráficos demasiado interativos (Allen et al., 2010).

  Rhodes é muito inspirado num estilo de *Framework*s MVC, como *Ruby* ou *Rails*. *Developers* que usem esta *Framework* não necessitam de se preocupar com o

[34] http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/ (ultimo acesso: 22/12/2013)
[35] http://www.rhomobile.com/wiki (último acesso: 20/12/2013)
[36] https://www.ruby-lang.org/pt/ (último acesso: 20/12/2013)

alojamento dos dados nem lógicas de sincronização, apenas precisam de se focar na apresentação da sua aplicação e da sua lógica de negócios (Allen et al., 2010).

- ***Titanium***[37] **-** *Appcelerator's Titanium Mobile platform* é uma *open source* de suporte comercial para desenvolver aplicações nativas multiplataforma com recurso a tecnologias web. *Appcelerator* introduziu esta plataforma em Dezembro de 2008, e recentemente, passou a estar funcional também para dispositivos *Blackberry*, o que anteriormente não era visível (Allen et al., 2010).

  Titanium consiste num SDK que oferece as ferramentas, compiladores e APIs necessários para criar e desenvolver aplicações, e também um ambiente visual para gerir os projetos do *developer* denominado *Titanium Developer*. Esta ferramenta proporciona um visual apelativo para o utilizador criar os seus projetos e permite que este utilize um editor de código à sua escolha para editar e criar código. Contudo, para criação de aplicações para diferentes plataformas, o SDK irá variar conforme o dispositivo a que se destinará a aplicação (Allen et al., 2010).

  Esta plataforma proporciona uma API independente para aceder a componentes nativas da interface do utilizador incluindo menus de navegação, caixas de diálogos, alertas, entre outros, assim como acesso a funcionalidades do dispositivo, como por exemplo, som, rede, base de dados local, etc.. O *developer* programa a sua aplicação em Javascript que depois será compilada em código nativo (Allen et al., 2010).

- ***AppMobi***[38] **–** "*Introducing appMobi, the mobile app development XDK for web developers. If you can build it for the web using* HTML*, CSS and JavaScript, now you can use appMobi to build it as an app for the iPhone, iPad and all Android smart phones and pads. Create robust, fully customized, 100% native API-compliant mobile apps in hours.*"[39].

  AppMobi é uma *Framework* que corre dentro de uma instância do Google Chrome, e acede a ficheiros de sistema locais do *developer* para criar as pastas e ficheiros do seu projeto. Oferece uma boa visualização das aplicações criadas pelo utilizador através do seu simulador (Miller, 2011).

  Em 2013, a empresa *Intel* adquiriu as ferramentas de desenvolvimento desta *Framework*. *AppMobi* suporta "*develop-once-deploy-everywhere*" em oito plataformas

[37] http://www.appcelerator.com (último acesso: 20/12/2013)
[38] http://www.appmobi.com/ - (último acesso: 20/12/2013)

e permite desenvolvimento local e em *cloud* pela *Intel's* XDK (Georgiev, Jana, & Shmatikov, 2014).

- ***JQTouch*[40]-** é uma *Open Source* inicialmente desenvolvida para aplicações web por David Kaneda. Trata-se de um Plug-in de *Jquery* ou *Zepto* para desenvolvimento web originalmente destinado a iPhones e Ipods (Allen et al., 2010). Atualmente é uma *Framework* que contém suaves animações, navegação e temas para *browsers Webkit* (iOS, *Android*, *Blackberry* e *WebOS*)[21]. Uma das vantagens desta ferramenta é a facilidade como transforma uma interface móvel elaborada em código HTML em algo semelhante a uma aplicação nativa (Allen et al., 2010).

  JQTouch é uma das *Framework*s pioneiras de Javascript. Proporciona interfaces de utilizador básicas. Sendo umas das primeiras ferramentas de Javascript a serem implementadas, os seus recursos são um pouco limitado relativamente às *Framework*s seguintes, contudo é bastante elegante (Miller, 2011).

- ***Jquery Mobile*[41] -** é um *Plug*-in construído no topo da Biblioteca de Jquery que suporta todos os dispositivos móveis mais populares. Tal como *jQTouch* esta *Framework* oferece várias interfaces de utilizador com controlos e animações ao seu próprio estilo. É uma *Framework* de tamanho bastante reduzido (Miller, 2011).

  Esta ferramenta foi desenhada principalmente para construir sítios web responsivos e aplicações que são acessíveis em todos os dispositivos móveis e desktop. Oferece navegação Ajax com transições de página, eventos de toque e vários *widgets*. O seu leve código é desenvolvido com *progressive enhancement*[42] (PE) e é detentora de um *design* flexível e simples[22].

  Jquery Mobile entrega boas experiências ao utilizador de aplicações web em dispositivos móveis, principalmente para interfaces de toque, multiplataformas e com recurso apenas a HTML5 como código padrão. Esta *Framework* não é muito apropriada para amantes de Javascript (exceto para tópicos mais avançados), contudo, é ótima para web *designers* que não usufruam de muita experiência a programar em linguagens como Javascript (Firtman, 2012).

---

[39] http://xdk.appmobi.com/ - (último acesso: 20/12/2013)
[40] http://jqtjs.com/ - (último acesso: 20/12/2013)
[41] http://www.jquerymobile.com – (último acesso: 20/12/2013)
[42] "***Progressive Enhancement*** *is a powerful methodology that allows Web developers to concentrate on building the best possible websites while balancing the issues inherent in those websites being accessed by multiple unknown user-agents. Progressive Enhancement (PE) is the principle of starting with a rock-solid foundation and then adding enhancements to it if you know certain visiting user-agents can handle the improved experience."* - (Dwyer, 2009)

***Sencha Touch*[43] -** é uma *Framework* Javascript destinada ao desenvolvimento de aplicações web para dispositivos baseados em eventos de toque. Foi produzida pela *Sencha* (*ExtJs*), e lançada em 2007. Combina a *Framework* de Javascript *ExtJs* com *jQTouch* e a Biblioteca de Javascript *Raphäel*[44] (Allen et al., 2010). Não está dependente da biblioteca de Jquery e é compatível com *Android*, *iOS, Blackberry, Windows Phone* e muito mais[24]. *Developers* ao utilizarem esta *Framework* necessitarão de grandes conhecimentos em Javascript para obterem vantagem desta (Allen et al., 2010).

É uma *Framework* orientada a objetos e tem uma curva de aprendizagem relativamente maior que as duas anteriores. Elementos visuais no DOM, são mais comuns serem criados via Javascript do que HTML, o que torna esta *Framework* mais apelativa para web *developers* mais avançados em programação (Miller, 2011).

c) *Frameworks* CSS

As *Cascading Style Sheets* oferecem ao programador o controlo sobre o design das suas páginas web. Num sentido metafórico, ao se utilizar CSS é como se estivesse a "*vestir*" um sítio web com *borders*, cores, backgrounds, etc. (McFarland, 2012).

As CSS trabalham juntamente com HTML. Enquanto HTML fornece a estrutura do documento, organizando a informação em cabeçalhos, parágrafos, listas, entre outros, as CSS preocupam-se em tornar esses parágrafos, cabeçalhos, mais visuais e com melhor aspeto.

As CSS são uma linguagem declarativa e não de programação (Clarke, 2013). As propriedades de estilo e valores que são declarados com as regras das CSS são exatamente o que o *browser* usa para "*pintar*" o ecrã. Desta forma, surgiram os pré processadores de CSS, pensados com o intuito de transformar o CSS numa linguagem mais simples e intuitiva, oferecendo compatibilidades entre *browsers* e alguma lógica para auxiliar na caracterização e desenhos de uma página web.

A SaSS[45] é uma linguagem que é utilizada para criar estilos num documento de forma clara e estruturada (Clarke, 2013). Proporciona uma sintaxe mais elegante e implementa vários recursos úteis para a criação de estilos manejáveis e lógicos. A SaSS adiciona regras, variáveis, *mixins* às CSS. Gera documentos bem formatados e estruturados tornando o código mais fácil de organizar e manter.

---

[43] www.sencha.com/products/touch - (ultimo acesso: 20/12/2013)
[44] http://raphaeljs.com/ - (ultimo acesso: 21/12/2013)
[45] http://sass-lang.com/ (27/12/2013)

A Less[46]é linguagem tem mais ou menos as características da anterior, na medida em que adiciona funcionalidades às CSS tais como, variáveis, *mixins* e outras funções para tornar as CSS mais sustentável. Less é executado dentro do *NodeJS* e no *browser*.

Por último, não menos popular, segue-se a linguagem *Stylus*[47] caracterizada pelas mesmas funcionalidades que as anteriores e proporcionando semelhantes vantagens.

Para além da existência de pré processadores para ajudar na melhoria da sintaxe de CSS, existem, também algumas *Framework*s para tornar essa tarefa ainda mais simples, são algumas delas:

- Compass
- Zurb Foundation
- Less *Framework* 4
- Yaml
- Blueprint
- Gumby
- Pure
- Bootstrap

Estas *Framework*s não são muito distintas nas suas funcionalidades e características, têm o mesmo objetivo de implementar CSS sustentável e de fácil organização para o utilizador. Muitas delas já vêm com o conceito de responsividade implementado como é o exemplo da *Framework* Zurb Foundation e *Bootstrap*, criando grelhas e esboços fluidos, adaptáveis às diferentes resoluções de ecrãs.

B. Critérios para escolha da *Framework* (mv*)

Como referido anteriormente, a grande variedade de *Frameworks* veio potenciar indecisões numa fase inicial de novos projetos web. É crucial escolher uma ferramenta que se enquadre nas necessidades do projeto, bem como da equipa de desenvolvimento. No processo de escolha de uma *Framework* deve-se ter em consideração diversos critérios, tais como, a qualidade, performance, comunidades ativas, manutenção, maturidade, entre outros. O processo de seleção de uma *Framework* e os respetivos critérios de decisão serão descritos nos tópicos seguintes.

[46] http://less.github.io/less-docs/ (27/12/2013)
[47] http://learnboost.github.io/stylus/ (27/12/2013)

1. Ferramentas e técnicas para auxiliar a seleção de uma *Framework*

Ao iniciar um novo projeto *front-end* javascript, um programador depara-se com uma vasta área de *Frameworks* que evitam a escrita de código puro em Javascript (embora este também possa decidir escrever código puro) (Graziotin & Abrahamsson, 2013). É necessário que dentro deste ramo de *Frameworks* seja escolhida uma que seja adequada ao seu projeto e que o tempo da sua seleção não reflita custos de produção e aprendizagem no desenvolvimento do projeto.

Existem, atualmente, dezenas de plataformas e documentos que disponibilizam aos programadores auxílio e prestação na comparação e caracterização das diferentes ferramentas, permitindo que estes aprofundem melhor os seus conhecimentos acerca destas e possam comparar e selecionar a(s) que melhor define(m) os seus projetos.

Foram selecionadas algumas ferramentas consideradas mais importantes e relevantes para este estudo. São elas:

- *todoMVC*: esta ferramenta já foi mencionada previamente, e serviu de base à escolha de algumas das *Framework*s Javascript subscritas. Esta plataforma contém maior parte das ferramentas que seguem mais ou menos o modelo MVC e fornece uma mini aplicação de afazeres com código elaborado por cada uma. Oferece liberdade ao utilizador de experimentar as *Framework*s na aplicação incentivando este a tirar as suas próprias conclusões sobre determinados critérios, tais como curva de aprendizagem, funcionalidades pertinentes ao objetivo do seu projeto, entre outros.
- *Front-end Framework*s v2.6[48]: Coleção comparativa de *Framework*s CSS com caracterização de cada uma e compatibilidades com os diferentes *browsers* e suas versões e os dispositivos móveis. Esta plataforma possibilita ao utilizador estudar se as suas decisões por algumas destas ferramentas influencia a escolha da *Framework* javascript ou vice-versa.
- *JSter*[49]: catálogo de Bibliotecas e *Framework*s de Javascript. Esta plataforma contém todas as possibilidades e soluções de Javascript para ajudar o utilizador encontrar uma que corresponda aos seus propósitos.
- *Stack Overflow*[50]: sítio web de perguntas e respostas para programadores profissionais. É uma plataforma que permite aos *developers* colocar as suas dúvidas

[48] http://usablica.github.io/front-end-*Frameworks*/compare.HTML (ultimo acesso: 26/12/2013)
[49] http://jster.net/ (ultimo acesso: 26/12/2013)
[50] http://www.stackoverflow.com (ultimo acesso: 26/12/2013)

enquanto desenvolvem os seus projetos, assim como dar-lhes liberdade para responder e ajudar outros programadores.

- *Mobile Frameworks comparison chart*[51]: sítio web com algumas *Framework*s mobile distinguidas por determinados critérios que ajudarão o utilizador a selecionar a que melhor emprega na sua aplicação. Disponibiliza também um pequeno questionário sobre alguns requisitos que o *developer* pretende para a sua aplicação e deste modo, aconselhar-lhe a ferramenta que contribui para esses requisitos.

É de salientar ainda, que esta fase de seleção faz parte do estado inicial de um projeto e deve ser uma decisão que ofereça consequências positivas ao longo de todo o seu desenvolvimento.

2. Critérios gerais

É fundamental, na procura de uma *Framework*, ter em consideração determinados aspetos técnicos ou de implementação que implicarão os resultados finais de uma aplicação web. Seguidamente serão apresentados alguns critérios gerais considerados também importantes a ter em reflexão na seleção de uma *Framework* (Graziotin, Abrahamsson, 2013; Osmani, 2013):

- **Maturidade**: novas *Framework*s aparecem geralmente constantemente rodeadas de novas atualizações que podem comprometer o projeto do programador. Uma aplicação madura não corre esse risco, para além de que contém uma documentação mais detalhada e informativa.
- **Tamanho**: é importante conhecer qual o tamanho real que a *Framework* implica, ou seja, o tamanho total incluindo dependências, minificação, *gzipping*, entre outras modularidades. Saber quantas dependências a *Framework* contém, é uma questão importante que o programador deverá considerar. Uma ferramenta pode ser enganosa relativamente ao seu tamanho, induzindo a um tamanho bastante pequeno, contudo, somando as suas dependências poderá transportar grandes pesos e problemas de performance para a aplicação.
- **Popularidade e comunidade**: é importante para uma *Framework* conter uma comunidade ativa que serve de auxílio a muitos programadores aquando o desenvolvimento das suas aplicações. É imprescindível a existência de programadores suficientes que utilizem esta *Framework* e que forneçam tutoriais, aplicações de referência, vídeos, entre outros métodos de assistência a utilizadores que pretendam começar um novo projeto com determinada ferramenta.

[51] http://www.markus-falk.com/mobile-*Framework*s-comparison-chart/ (ultimo acesso: 26/12/2013)

- **Funcionalidades**: saber do que a *Framework* é capaz é um dos aspetos mais importantes na sua seleção. É importante despender de algum tempo na análise do código fonte e funcionalidades da ferramenta para perceber se são adequadas para os requisitos do projeto.
- **Produção e manutenção**: importa ao utilizador conhecer se a ferramenta já foi provada em produção, se existem no mercado aplicações acessíveis desenvolvidas com a *Framework*. É importante analisar o seu portfólio. Saber só se uma *Framework* existe em produção ou não, é insuficiente, torna-se essencial ser capaz de olhar para aplicações reais e ficar inspirado pelo que a *Framework* mostra que é capaz.
- **Experiência**: conhecer apenas as funcionalidades de uma *Framework* não é suficiente para concluir que é a ideal. É aconselhável a um programador criar alguma prática com a mesma para compreender se esta lhe é acessível e de fácil compreensão para o seu trabalho.
- **Flexibilidade**: *Framework*s pouco flexíveis bloqueiam ou sugerem ao utilizador desenvolver funcionalidades de uma maneira específica e pré-definida, senão mesmo a maneira destas.
- **Documentação**: é essencial conhecer as funcionalidades e os componentes da *Framework*. Estes aspetos devem estar acompanhados por uma boa documentação, com bastante detalhe, servindo de guia e tutorial aos *developers*.
- **Qualidade**: este facto é expresso em termos de certas medidas tais como linhas de código, comentários, complexidade, manutenção, entre outros aspetos que são importantes ter em conta ao comparar certas *Framework*s.
- **Curva de aprendizagem**: Anteriormente foi referido que o utilizador deverá ter algum contacto com a ferramenta, para concluir se é adaptável ao seu trabalho ou não. Este facto reflete-se também na curva de aprendizagem. O programador ao experimentar uma *Framework* poderá levar muito ou pouco tempo a interiorizar os objetivos e requisitos desta.
- **Suporte**: atualmente é alargada a variedade de *browsers* existentes. Embora uns sejam mais utilizados e populares que outros, não é aconselhável descartar os menos populares. Com certeza existirá uma pequena percentagem de utilizadores finais ainda a utilizar essas plataformas. Desta forma, é importante que uma *Framework* ofereça suporte para todos ou a maioria dos *browsers*.

3. Compatibilidade

A compatibilidade é um dos critérios mais importantes na escolha de uma *Framework* Javascript MVC. Para que serve selecionar uma *Framework* que não está disponível para a maioria dos *browsers*? De que vale escolher uma *Framework* se esta não é compatível com a ferramenta móvel ou CSS selecionada? Já tenho o meu *back-end* definido e a ligação dos dados para apresentar ao cliente planeados, a *Framework* que escolhi não é compatível com estes métodos, mudo a lógica do projeto ou escolho uma nova *Framework*? Estas questões são bastante pertinentes no plano de escolha de uma ferramenta *front-end client-side* javascript. A *Framework* selecionada terá de preencher os requisitos do projeto estipulados pela empresa/programador. O utilizador poderá optar por diferentes caminhos, tais como, escolher primeiro qual a *Framework* adequada às funcionalidades da sua aplicação e posteriormente observar quais outras ferramentas móveis, CSS, ligação de dados, serão compatíveis com a primeira *Framework*. Poderá também já ter todo o seu projeto definido, incluindo tecnologias, e por último se preocupar com a *Framework* javascript. Nesse caso a escolha desta será possivelmente mais dificultada, tendo o programador de testar e experimentar as tecnologias já selecionadas com a nova ferramenta e concluir se estas são adaptáveis e compatíveis, trabalhando mutuamente para o melhor do projeto.

Anteriormente, neste documento, foram caracterizadas algumas *Framework*s de CSS e móveis que poderão servir de opção para um programador. Estas ferramentas terão de se compatibilizar entre si e com a *Framework* javascript MVC que o utilizar irá escolher ou já terá selecionado. Problemas de compatibilidade que poderão surgir com a junção destas ferramentas destacam-se principalmente nos ficheiros javascript. Estes ficheiros poderão incluir funções que colidem entre si, resultando em conflitos que o programador não conseguirá resolver, visto serem ficheiros pré concebidos e maior parte das vezes minificados. O sistema de Templates de cada *Framework* será variado entre elas, com funcionalidades idênticas que possibilitarão mais colisões de código. É imprescindível ao programador estudar bem estes casos para conseguir tirar o maior número de vantagens das diferentes *Framework*s existentes.

O *Front-end* e *Back-end* comunicam entre si através de diversas tecnologias que servem de ponte entre estes e transportam os dados bidireccionalmente. Uma *Framework* que possua métodos de receção e envio dos dados bastante eficazes e que facilitem a comunicação entre o cliente e servidor será a mais eficaz para o programador. É importante a compatibilidade e a adaptabilidade que a *Framework* oferece para

tecnologias como JSON, *Representational State Transfer* (REST), XML, *Simple Object Access Protocol* (SOAP) entre outras (Runeberg, 2013).

Ter um *back-end* que serve *JSON* abre bastantes possibilidades para a integração em aplicações móveis. Pedidos em *JSON* requerem menos dados para serem transferidos, o que pode ser visto com grande benefício, principalmente na ocorrência de operações limitadas, como por exemplo uma baixa banda larga, residente maioritariamente em redes móveis (Runeberg, 2013). O *JSON* tem-se tornado bastante popular na divulgação de APIs. Contém um padrão de modelos de armazenamento para bases de dados que não utilizem *Structured Query Language* (noSQL) e torna especialmente simples armazenar dados com profundas dependências (Runeberg, 2013).

"JavaScript Object Notation, or JSON, is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 19993. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."[52]

Os serviços web disponibilizam formas de diferentes tipos de aplicações (entre várias plataformas e sistemas operativos) interagirem (Moraes, Breda, Gil, & Medaglia, s.d.). Foram criados, inicialmente, como uma interface padronizada baseada em tecnologias como XML e protocolos HTTP. Foi desenvolvida pela *World Wide Consortium* (W3C)[53] e de acordo com esta, um serviço web é um sistema de *software* responsável por proporcionar a interação entre duas máquinas através de uma rede. A *Web Service Description Language* (WSDL) possibilita essa interação permitindo que sistemas interajam com um serviço web usando esta interface e enviando mensagens SOAP ou protocolos REST, entre outros (Moraes, Breda, Gil, & Medaglia, s.d.).

O XML é um padrão para a formatação de dados, uma maneira de organizar informações. Estes documentos podem ser facilmente compreendidos por programadores facilitando o desenvolvimento de aplicações compatíveis. Foi também desenvolvido pela W3C para oferecer à web uma forma simples de combater as

[52] json.org (último acesso: 27/12/2013)
[53] "*…um consórcio de empresas de tecnologia que tem como o objetivo criar padrões comuns para conteúdo da Web, apoiada por grandes empresas como a Microsoft, IBM, HP entre outras*" - (Moraes et al., s.d., p.1)

limitações inerentes do HTML e permitir novos tipos de aplicações para a internet. É um padrão de armazenamento de dados em formato de texto simples, podendo deste modo ser acedido em qualquer máquina ("A linguagem XML," s.d.).

As mensagens SOAP são documentos XML serializados seguindo o padrão W3C e enviados em cima de um protocolo de rede. Segundo a W3C esta interface é destinada essencialmente para comunicação entre aplicações, com um formato para envio de mensagens através da internet. É uma plataforma e linguagem independentes baseadas, como já foi mencionado, em XML. É uma linguagem simples e extensível.

Por outro lado, mensagens REST definem um conjunto de princípios arquiteturais, por onde se desenha um serviço web focado nos recursos de um sistema, incluindo como os estados desses recursos são endereçados e transferidos através de HTTP (Rodriguez, 2008). REST tornou-se bastante popular pelos seus sistemas distribuídos, onde cada transação tem de incluir informação suficiente sobre o estado do cliente que não existe no servidor, tornando possível a cada pedido, estes serem servidos por diferentes sistemas (Runeberg, 2013).

As Interfaces REST têm vindo a substituir as SOAP, pelo simples facto de a primeira ser mais simples e com uma estrutura mais legível. REST tem vantagens também ao utilizar os padrões de HTTP como GET, POST, DELETE e PUT (Runeberg, 2013).

4. Responsividade

O *Responsive* web *design* sugere que design e desenvolvimento de aplicações devem responder ao comportamento do utilizador e ao tamanho dos ecrãs, plataformas e orientação (Knight, 2011) como demonstrado na Ilustração 3. Esta prática consiste numa mistura de grelhas e esboços flexíveis, imagens e um uso inteligente de CSS *media queries*[54] para o desenvolvimento de aplicações e páginas que se adaptem a diferentes dispositivos, tais como *smatphone, tablets,* computadores portáteis, entre outros.

"The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the

[54] "*A **media query** consists of a media type and at least one expression that limits the style sheets' scope by using media features, such as width, height, and color. Media queries, added in CSS3, let the presentation of content be tailored to a specific range of output devices without having to change the content itself.*" - https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries (ultimo acesso: 28/12/2013)

fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must 'accept the ebb and flow of things.'[55]

O sítio web deverá automaticamente adaptar-se às novas resoluções. Em suma, o sítio web deverá conter tecnologia suficiente para automaticamente responder às preferências do utilizador. Isto irá eliminar a necessidade de criar diferentes desenhos e fases de desenvolvimento para cada novo dispositivo no mercado (Knight, 2011). O *Responsive* design transformou a maneira de pensar o design. É uma das soluções técnicas para programar um *site* para que elementos que o compõem se adaptem automaticamente à largura do ecrã do dispositivo em que está a ser visualizado.

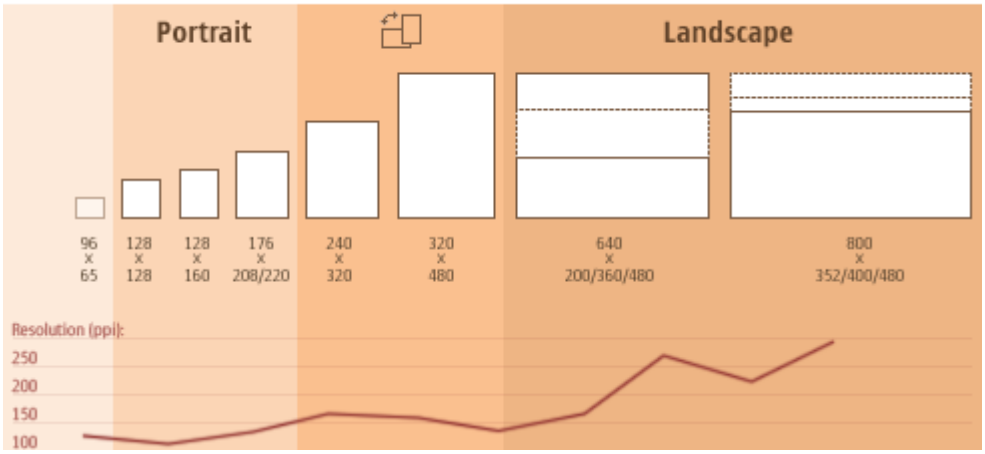


Ilustração 3 - diferentes resoluções e orientação dos dispositivos

A maior desvantagem de um sítio web responsivo é o desempenho (Sarraf, 2013). Este tipo de *sites* têm tendência a ser mais lentos, uma vez que todo o conteúdo é arquivado na página sendo mostrado ao utilizador apenas o necessário, dependendo do dispositivo em que se encontra (Sarraf, 2013). Por outro lado, o facto de apenas se construir um sítio web para múltiplos dispositivos acarreta vantagens não só para a experiência do utilizador, como também para as *search engine optimization* (SEO), por apenas um URL ser observado e desta forma garantir que toda a informação é transmitida para estas ferramentas, e não apenas algum conteúdo no caso de sítios web separados (Sarraf, 2013).

É importante para o utilizador perceber quais são as suas necessidades e compreender que os requisitos de um desktop são diferentes de um dispositivo móvel (Marcotte, 2011). O programador e a empresa terão de assumir a responsabilidade por

[55] - http://alistapart.com/article/dao (último acesso: 20/12/2013)

um sítio web responsivo para múltiplos dispositivos ou separar *mobile* de desktop tendo essa distinção dada à responsabilidade de um servidor. Ambos trazem vantagens que corresponderão mais ou menos aos requisitos do projeto da empresa, assim como, influenciarão as suas escolhas para a *Framework* javascript mvc.

5. Performance da aplicação

Inicialmente, neste documento foi relatado o quão importante era a performance de uma aplicação, que o impacto positivo de compromisso dos utilizadores com esta deve-se especialmente a melhorias na latência e performance da aplicação.

O autor Steve Souders (2007) define catorze regras para aumentar a performance de uma aplicação, são elas:

1- Reduzir o número de pedidos HTTP
2- Utilizar um *content delivery network* (CDN)
3- Adicionar expiração nos cabeçalhos
4- Compactar componentes
5- Colocar os estilos no topo do documento
6- Evitar expressões de CSS
7- Mover o código javascript e CSS para ficheiros externos
8- Reduzir pesquisas de DNS
9- Minificar o javascript
10- Evitar redireccionamentos
11- Remover *script* repetidos
12- Configurar as ETags
13- Tornar o Ajax armazenável em cache

Existem diversos motivos que alteram a performance das aplicações, contudo uma aplicação com boa manutenção e organizada é um dos primeiros passos para melhorias de performance. Um código fonte otimizado e minificado, redução do número de dependências, cálculo do peso das imagens e possibilidades de redução, entre outros, todos estes aspetos fazem parte de uma primeira abordagem na melhoria da qualidade de um sítio web.

Uma reduzida banda larga pode ser um dos principais fatores que diminui a performance de um sítio web. Cada vez mais o uso de dispositivos móveis se torna

visível, trazendo larguras de banda mais baixas que a versão desktop obrigando as aplicações a adaptarem-se a estas mudanças.

É relevante ter um servidor devidamente configurado e com as características adequadas ao tipo de utilização que irá ter, por exemplo, estar preparado para grandes quantidades de informação, preparado para múltiplos acessos em simultâneo, entre outros aspetos. A falha de um sistema destes ou lentidão durante o acesso dos seus utilizadores poderá ter custos inevitáveis, como por exemplo, a nível das desistências dos visualizadores.

Finalmente, na escolha da *Framework* mvc, é importante ter em conta os custos que esta terá para a performance do sítio web, a nível de memória, dependências, peso, linhas de código, entre outros critérios que deverão ser analisados pelo utilizador, por forma a selecionar a que ofereça mais vantagens para a performance do seu sítio web.

a) Algumas ferramentas para auxiliar na performance

Como exemplo de ferramentas de técnicas para análise da performance, destaca-se:

- *Lo*-Dash: é uma biblioteca que oferece consistência, performance e muitas mais funcionalidades para ajudar os programadores a otimizar os seus sítios web;
- *Garbage Collection:* o Javascript, normalmente, utiliza esta ferramenta para poupar memória usada por *strings,* objetos, arrays e funções que não estejam mais em uso. Esta ferramenta tem como principal vantagem a fácil compreensão para determinar quando é seguro recuperar memória (Flanagan, s.d.).
- *Web storage*: Com o surgimento do HTML, armazenar dados localmente tornou-se mais acessível e uma boa solução para melhorias na performance nas aplicações. É uma forma segura e rápida e permite alocar grandes quantidades de dados sem prejudicar a performance do sítio web[56].

6. Economia de custos de produção

Selecionar a ferramenta mais adequada a um projeto nem sempre é tarefa fácil. É necessário conhecer primordialmente os requisitos do projeto e os objetivos da empresa para se prosseguir à seleção das *Frameworks* que se enquadrem na metodologia elaborada.

Uma má decisão poder retornar em consequências negativas para os programadores e gestores de projeto É útil elaborar uma estimativa de custos para o desenvolvimento do projeto e criar meios para contorná-los.

Segundo os autores Grahame Steven (s.d) a curva de aprendizagem reflete-se numa importante técnica para prever quanto tempo irá levar a empreender novas tarefas e, neste caso, empreender os objetivos de uma *Framework*. Este tempo irá afetar ou melhorar o rápido e eficaz desenvolvimento de um projeto, comprometendo tanto o programador como o gestor de projeto.

Terminado o enquadramento de base teórica que influenciou as restantes fases do desenvolvimento desta investigação segue-se, nos próximos capítulos, a descrição da metodologia e de todo o processo de seleção da *Framework mvc* de javascript.

[56] http://www.w3schools.com/HTML/HTML5_webstorage.asp - (ultimo acesso a 20/12/2013)

III. METODOLOGIAS DE INVESTIGAÇÃO

A. Metodologia escolhida

No desenvolvimento desta investigação optou-se por seguir uma metodologia mista de investigação-ação, a qual consiste em aumentar o conhecimento geral e a produzir resultados que possam ser usados em tomadas de decisão ou melhoria de programas (Coutinho et al., 2009).

Segundo (Coutinho, 2011) as principais características desta metodologia são:

- Participativa e colaborativa;
- Prática e interventiva;
- Cíclica;
- Critica;
- Auto avaliativa;

Da análise das principais características da metodologia de investigação com as características deste estudo, verificou-se que o caracter participativo e colaborativo desta metodologia se refletiu nesta investigação através do processo de recolha de dados não ter sido baseado apenas em consulta de artigos, mas também na recolha de opiniões de programadores, uma vez que estes têm um melhor e mais aprofundado conhecimento da realidade. O segundo, prático e interventivo, uma vez que este estudo não se limitou só a descrever a realidade, mas também a intervir nela própria, ou seja, pode influenciar a decisão de escolha de *Frameworks* nas empresas e nos seus programadores e gestores. Cíclico, uma vez que foi um estudo onde as descobertas iniciais podiam gerar possibilidades de mudança, e após implementadas e avaliadas serviriam de introdução ao clico seguinte, por exemplo, a avaliação das *Frameworks* pelos vários critérios contribuiu para um processo cíclico, no sentido em que uma *Framework* que mostrasse vantagem segundo um determinado critério poderia contribuir para uma reavaliação das restantes e assim sucessivamente. Não se tratou de uma investigação crítica uma vez que, a comunidade de participantes apenas procurou melhorar as suas práticas e não, eles próprios, terem servido como agentes de mudanças. Finalmente, este estudo revelou-se auto avaliativo, uma vez que as *Frameworks* foram continuamente avaliadas segundo os critérios escolhidos, procurando adaptar e produzir novos conhecimentos ao público-alvo.

Para este estudo foi necessário adotar determinados caminhos para se poder obter os resultados esperados. Esses caminhos consistiram em planear, atuar, observar e refletir no sentido de conduzir a melhorias no processo de seleção da *Framework* mais

adequada para um projeto. O objetivo desta investigação não foi tanto o de gerar conhecimento aos programadores, mas sim, de questionar as suas práticas e os valores que as integram com a finalidade de os explicar e melhorar. Segundo o autor Simões (1990) apud (Coutinho et al., 2009, p.9) "*o resultado da investigação terá sempre um triplo objetivo: produzir conhecimento, modificar a realidade e transformar os actores*", o que, para este estudo, se baseou em apresentar os conceitos e características das várias *Frameworks* existentes e dos critérios para a sua escolha, seguida de uma análise e caracterização dessas tecnologias segundo os critérios assinalados e, finalmente, a criação de caminhos de escolha possíveis para garantir respostas aos programadores e gestores de projeto.

B. Modelo de análise

Após a definição dos objetivos desta investigação avançou-se para a construção do modelo de análise. Como principais conceitos foram identificados os seguintes: *Frameworks Web* e *business-to-business*. Partindo destes conceitos tentou-se identificar as dimensões e os indicadores associados a cada conceito. O modelo de análise completo encontra-se representado na Tabela 1.

O primeiro conceito de *Framework Web* abrange um conjunto de ferramentas e tecnologias bastante alargado e que se encontra em contínuo crescimento. Este crescimento reflete-se na constante atualização e no aparecimento de novas *Frameworks*. Esta evolução pode potenciar, nas empresas de vertente tecnológica e nos programadores, indecisões no arranque de um projeto. Indecisões que podem levar a grandes custos caso estas não sejam as melhores. É necessário considerar e analisar todas as possibilidades tecnológicas onde estas *Frameworks* podem ser aplicadas, seja *mobile*, HTML, CSS, javascript, o seu próprio modelo de funcionamento e características, sendo que, todos estes fatores influenciam o processo de seleção de uma *Framework*.

O segundo conceito de *business-to-business* já remete para o mercado-alvo da aplicação que se encontra a ser desenvolvida no âmbito desta investigação. É um conceito direcionado para a vertente mais económica e para um estudo de aplicações *mobile* e web nesta área de negócios.

Conceitos	Dimensões	Indicadores	Instrumento(s) de recolha de dados
***Framework*s web**	Modelo de funcionamento	*Client-side* *Front-end*	Análise de documentos
	Características	*Spaghetti code* modularidade	Análise de documentos Entrevista
	*Framework*s Mobile	Tipo de aplicações Maturidade Responsividade Performance Economia de custos Técnicas de seleção	Análise de documentos Entrevista
	*Framework*s HTML/CSS	Maturidade Responsividade Performance Economia de custos Técnicas de seleção	Análise de documentos Entrevistas
	*Framework*s Javascript	MVC Maturidade Responsividade Performance Economia de custos Técnicas de seleção	Análise de documentos Entrevistas
Business-to-business	Económica	Tipo de negócio	Análise de documentos
	Aplicações	Web mobile	Análise de documentos

Tabela 1 - Modelo de Análise

C. Recolha de dados

Segundo os autores Quivy e Campenhoudt (1995) no processo de recolha de informação é essencial seguir alguns passos, tais como, a conceção de um instrumento de observação capaz de produzir todas as informações adequadas e necessárias para testar as hipóteses, posteriormente será necessário testar esse instrumento e por fim recolher os dados através desse meio.

A recolha de dados para este estudo incidiu essencialmente nas perceções e interpretações da experiência de programadores *front-end client-side* e gestores ao usarem, selecionarem ou preferirem determinada tecnologia, quais as suas opiniões e *feedback* ao usarem a ferramenta, as suas preferências, a interpretação que têm acerca dos critérios para a escolha da mesma, assim como o(s) caminho(s) que leva(m) para a sua seleção. Deste modo, consideraram-se duas técnicas adequadas para a recolha de informação, são elas: a análise de documentos e a entrevista.

A análise de documentos é uma técnica que implica uma pesquisa e leitura aprofundadas de artigos escritos. Esta técnica serviu, essencialmente, para um estudo mais completo e fidedigno da recolha dos dados, uma vez que incidiu, principalmente, em opiniões ou conselhos de profissionais programadores e gestores que tenham experienciado algumas das *Frameworks* que foram estudadas e caracterizadas, assim como, todo o seu processo de seleção. Está técnica de análise de documentos serviu também como plano de contingência para o caso de a segunda, que será descrita no parágrafo seguinte, ter falhado ou não se ter desenvolvido como previsto.

A segunda técnica considerada essencial a esta investigação, como referido, foi a entrevista. Segundo os autores Quivy e Campenhoudt (1995), a entrevista é caracterizada como sendo *“ (...), uma verdadeira troca, durante a qual o interlocutor do investigador exprime as suas perceções de um acontecimento ou de uma situação, as suas interpretações ou as suas experiências, (...)* ”. A descrição dada pelos autores descreve o objetivo primordial da recolha dos dados que é, como citado anteriormente, recolher os relatos das experiências dos entrevistados relativamente ao uso de algumas *Frameworks* javascript.

Para a realização da entrevista foi necessário ter em atenção que a investigação em causa abordava uma vasta área de conceitos, e por isso colocar questões demasiado objetivas e fechadas poderia não resultar no que era pretendido, cada pessoa tem uma experiência e opinião diferente da outra, portanto, era mais simples e eficaz se essas experiências forem descritas de forma livre e sem um caminho de conversa demasiado específico, impedindo o participante de abordar todas as suas perceções. Foi importante,

por fim, ter um caminho de entrevista mais ou menos definido e não demasiado objetivo para que o foco principal da conversa não se perdesse. Pelos motivos anteriores, optou-se por seguir a entrevista do tipo livre ou semi-estrutura, ou seja, no momento de a colocar em prática apenas se dispôs de tópicos relativos ao tema em estudo, os quais foram abordados ao interlocutor de forma livremente escolhida no momento e de acordo com o desenrolar da conversa.

Num processo de recolha de dados por entrevista é importante especificar o perfil dos participantes a inquirir para se poder obter respostas claras e acordantes com os objetivos da investigação. É necessário conhecer e delimitar as características que se pretendem num participante e, para este estudo, foi importante que o perfil do interlocutor fosse compatível com o de um programador profissional ou gestor, e que este fosse entendedor dos conceitos e das dimensões referidas no ponto anterior para que conseguisse responder às questões colocadas abertamente e de forma acertada.

Finalizado todo o processo de escolha da *Framework* mvc de javascript, foi realizada, por último, uma entrevista por forma a validar a *Framework* selecionada. A entrevista foi abordada por alguns elementos da empresa Prodcent. Era importante obter as experiências e opiniões dos vários elementos da empresa, por forma a validar se a decisão tinha sido bem-sucedida e não tinha causado custos extras.

D. Etapas do estudo

Nos próximos parágrafos serão apresentados os resultados da análise de todos os dados recolhidos, assim como todo o processo que levou à *Framework* final e posterior validação da mesma. Na Ilustração 4 pode-se visualizar esquematicamente todas as etapas deste estudo que conduziram esta investigação e que serão detalhadas seguidamente.

Primeiramente foram elaboradas algumas entrevistas a programadores e gestores de projetos de forma a conhecer opiniões e critérios de escolha de uma *Framework* nos seus ambientes de trabalho. Essas entrevistas foram posteriormente analisadas e deram resultado aos principais critérios de escolha que fizeram parte da análise das *Frameworks* também recolhidas nessa análise dos dados das entrevistas.

Como o processo de escolha da *Framework* javascript em contexto empresarial decorreu antes da documentação da seleção dos principais critérios para a escolha de uma *Framework* javascript mvc, achou-se pertinente, como mencionado anteriormente, descrever todo o procedimento efetuado para a escolha da *Framework* na empresa Prodcent de forma a validar e concluir se essa decisão atualmente ainda é plausível e corresponde aos resultados finais desta dissertação. Os resultados deste processo foram, depois, reunidos com os dados retirados das entrevistas para conclusões mais detalhadas e comparáveis.

Após a recolha dos dados das entrevistas reunidas com o processo de escolha da *Framework* em contexto empresarial foi elaborada uma avaliação de todas as *Frameworks* mencionadas no resultado da reunião anterior. Estas foram destacadas de acordo com a frequência com que foram mencionadas ao longo das análises.

Seguidamente, as *Frameworks* selecionadas foram avaliadas de acordo com os critérios de seleção identificados no processo de análise dos dados das entrevistas. Foi elaborada uma tabela de pontuações entre zero e três e, a cada *Framework* foi dada uma classificação segundo o critério em análise. Posteriormente, essas pontuações foram incrementadas por forma a obter uma pontuação final de cada *Framework* e deste modo, chegou-se à *Framework* que mais correspondeu aos critérios selecionados e que era ideal para o projeto em contexto empresarial. Embora os critérios de seleção fossem importantes, importava conhecer quais os principais requisitos e critérios inerentes ao projeto e selecionar uma *Framework* que fosse viável às funcionalidades que o projeto propunha. Por este motivo, a *Framework* selecionada da tabela dos critérios foi validada segundo os principais requisitos ao projeto, onde foi duplicada a sua pontuação por cada critério. Este processo repetiu-se para todas as *Frameworks* por forma a verificar se os

resultados anteriores variavam significativamente ou alteravam a *Framework* selecionada anteriormente.

Os resultados finais do procedimento anterior poderiam não coincidir com a *Framework* que já havia sido selecionada para o projeto em contexto empresarial, contudo, não significaria que a *Framework* não fosse viável ao mesmo. Por forma a validar a afirmação anterior foram efetuadas, por último, entrevistas aos programadores e gestores do projeto e que tivessem conhecimento da *Framework* em uso. Foi dado um feedback por estes acerca dos principais aspetos positivos e negativos da *Framework* para o projeto e da utilidade desta para o trabalho a decorrer. Por último, essas entrevistas foram analisadas e descritas as principais conclusões desses resultados e de todo o procedimento de escolha de uma *Framework* javascript mvc.

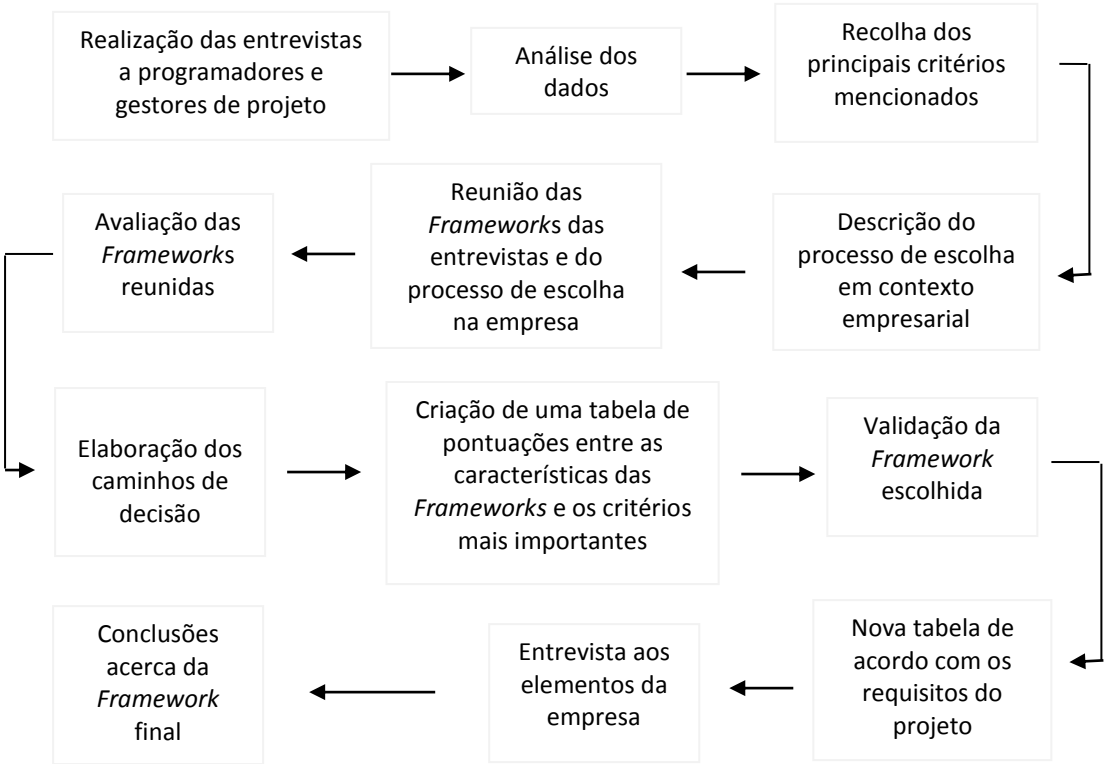


Ilustração 4 - Etapas do estudo

IV. RECOLHA E ANÁLISE DOS DADOS

A. Análise das entrevistas

Inicialmente, nesta dissertação, na Tabela 1 foram apresentadas as várias componentes pelas quais se pretende obter resposta à problemática em questão neste estudo. Um método utilizado para a obtenção destas respostas foi a entrevista exploratória, uma vez que se procurava conhecer as perceções e interpretações da experiência de programadores e gestores de projeto no processo de seleção e avaliação das tecnologias front-end client-side para os seus projetos.

O tipo de entrevista exploratória utilizado para a recolha de dados foi a entrevista livre ou semiestruturada, visto esta poder ser discursada de forma livre e sem objetivos demasiados específicos, concedendo liberdade ao entrevistado de relatar as suas experiências. Foi elaborado um guião de entrevista (Anexo 1 - Guião de entrevista) para servir de acompanhamento ao interlocutor durante a entrevista por forma a obter alguma organização no desenrolar da mesma. A entrevista procurou abranger todo o processo que envolvia a escolha dos critérios e da *Framework*, desde quem realizou essa escolha até ao tempo despendido para chegar a uma conclusão final. Finalmente, como se tratou de uma investigação do tipo qualitativa a análise dos dados foi maioritariamente de conteúdo, não havendo necessidade de obter quaisquer resultados estatísticos.

Foram realizadas cinco entrevistas que tiveram como fonte de informação três programadores e dois gestores de projeto. Como sistema de registo de informação foi utilizado maioritariamente o áudio, havendo apenas um caso em que a entrevista foi realizada por escrito. A experiência, neste tipo de decisões, acerca de quais as melhores *Frameworks* para a concretização dos projetos de cada entrevistado é distinta. Este aspeto verifica-se principalmente devido à grande variedade de *Frameworks* existentes na atualidade que podem incrementar o tempo até à decisão final, bem como o número de indecisões entre equipa. Na Tabela 2 foram caracterizados os entrevistados segundo a área de formação, o vínculo profissional e são ainda apresentados os principais projetos em que estiveram envolvidos. A cada entrevistado foi dado uma referência empírica para futuras citações ao longo do presente documento.

Entrevistado	Área de formação	Empresa	Projetos e tipologia	Plataformas
(E1)	Professor do Departamento de Comunicação e Arte na área das Novas Tecnologias e coordenador do projeto Sapo Campus.	Sapo	Projetos web com a preocupação de responder a uma audiência muito grande.	Projetos baseados em web mas que possam responder a todas as plataformas.
(E2)	Estudante de Mestrado em Comunicação Multimédia e web developer.	Ubiwhere	Projeto web de gestão dos horários e trabalhadores da empresa.	Desktop
(E3)	Responsável pela experimentação e harmonização tecnológica do departamento de Coordenação tecnológica.	PT Inovação e sistemas	(Sem dados)	(Sem dados)
(E4)	Professor de Novas Tecnologias e sócio gerente da empresa Others - Digital Thinking	Others - Digital Thinking	Projetos em flash, várias campanhas publicitárias para meo, zon, tmn. Sítios web para a empresa Azeites andorinha. Recentemente, projetos em HTML para dispositivos móveis. Projetos em flash começaram a migrar para novos ambientes de HTML, css e javascript.	Projetos para a maior parte das plataformas.
(E5)	Informático em Comunicação Multimédia. Web developer.	Ubiwhere	Planify (rede social de desporto), Drapn (repositório de fichas documentais de agricultura e pescas do Norte)	Web, iOS, Android, mobile

Tabela 2 - Categorização dos entrevistados

Na análise dos dados das entrevistas tentou-se identificar as *Frameworks* mais utilizadas pelos entrevistados e os motivos que os conduziram à utilização de tais ferramentas na sua programação front-end. Seguidamente, era importante conhecer como se desenrolou o processo de escolha dessas *Framework*s por cada indivíduo e qual a ordem dos procedimentos. Dentro da descrição do processo de escolha procurou-

se obter informação acerca de quem fez a escolha, se pelo(s) gestor(es) ou pelo(s) programador(es) do projeto, quais os critérios utilizados para essa seleção e com base em que requisitos foram escolhidos esses critérios. Uma decisão deste género pode corresponder favoravelmente às expetativas de maneira a que não atrapalhe toda a estratégia do projeto ou implique alterações nas decisões anteriormente tomadas. Deste modo, procurou-se saber se haviam ocorrido dificuldades, e caso se obtivesse uma resposta positiva conhecer quais os contratempos sentidos na seleção da *Framework* para o desenvolvimento dos projetos dos entrevistados, principalmente devido à vasta abrangência de *Framework*s disponíveis atualmente. Embora nem todas as respostas à questão anterior tenham sido positivas, os entrevistados também foram interrogados acerca da utilização de alguma tecnologia de auxílio à seleção da *Framework*, e deste modo, achou-se pertinente tentar identificar o uso dessas tecnologias e quais. Por último, ainda referente à descrição do processo de escolha, procurou-se conhecer e estimar o tempo despendido desde a seleção dos critérios até à decisão final. Terminado o processo e a decisão final tentou-se saber se este tipo de escolhas costuma ser unânime na equipa ou se existem alguns desacordos que obriguem a recomeçar todo o processo.

Durante a análise de dados tentou-se identificar não só *Frameworks* de javascript, mas também as de CSS e para dispositivos móveis com o intuito de saber se essas *Frameworks* tiveram alguma influência na utilização da *Framework* de javascript escolhida anteriormente. Seguidamente, após mencionadas todas as ferramentas para o código front-end procurou-se obter informação acerca da ordem de decisões destas *Frameworks* (desde CSS até à ligação dos dados entre front-end e back-end) e se esta disposição teve, igualmente, alguma influência na *Framework* javascript escolhida, obrigando a renovações na metodologia da empresa.

Após os aspetos mencionados anteriormente o discurso direcionou-se mais para a experiência dos entrevistados e da equipa ao utilizarem e aprenderem a *Framework* e para a satisfação dos mesmos acerca dessa experiência. Procurou-se obter mais detalhes acerca da prática de utilização e nível de conhecimento dos inquiridos sobre a *Framework* selecionada, incluindo o tempo de aprendizagem da mesma, e se essa experiência é ou foi satisfatória. No final da entrevista perguntou-se aos entrevistados se estes desejavam acrescentar mais algum aspeto para além dos anteriormente mencionados.

B. Análise dos dados

Durante o processo de análise dos dados das entrevistas tentou-se identificar quais as *Framework*s mais destacadas. Assim, foi possível identificar as seguintes *Framework*s, bem como a sua frequência de uso:

- Jquery (3)
- AngularJs (2)
- KnockoutJs (1)
- EmberJs (1)
- Dart (1)
- Prototype (1)
- YUI (1)
- Twin Max (1)

Das *Framework*s recolhidas pode-se observar que Jquery foi a ferramenta que a maioria dos participantes mencionou seguindo-se a AngularJs.

Seguidamente procurou-se conhecer os motivos que estão na base da escolha de uma *Framework* para o código front-end. As motivações apresentadas pelos entrevistados foram diversas, cada entrevistado enunciou um motivo diferente. Dois motivos mencionados foram que a experiência e a tipologia do projeto são aspetos muito importantes e que influenciam a decisão final pela escolha de uma *Framework*, como por exemplo, projetos que impliquem animações requerem requisitos diferentes de websites para manutenção de dados. Acrescenta-se a estas razões anteriores o facto de o javascript estar a evoluir e a ganhar mais preponderância em aplicações web e móveis, existindo mais código a correr no cliente em vez de ser tudo em páginas estáticas que são geradas por um servidor e, por fim, também, pelo facto de a *Framework* ser compatível com vários dispositivos.

Relativamente à questão de quem é o responsável pelas decisões sobre as ferramentas front-end verificou-se que as respostas foram semelhantes. A maioria dos entrevistados mencionou que foi dada liberdade ao programador para escolher a *Framework* que queria programar, contudo essa decisão teria de passar pelo gestor ou por um responsável superior, e só após concordância de ambas as partes se prosseguia para o desenvolvimento do projeto com essa ferramenta. Apenas num caso essa decisão é tomada apenas pelos programadores. Desta forma, pode-se concluir que a escolha de *Framework*s é decidida pelo gestor de projeto, sugerida pelo programador e aprovada pelo gestor.

Após terem sido identificados os principais motivos para o uso de *Framework*s javascript no código front-end client-side, foi pedido aos entrevistados que descrevessem o processo de escolha dessas *Framework*s. Da análise dos dados das entrevistas verificou-se que as respostas foram bastantes diversificadas, sendo estas descritas nos parágrafos seguintes.

A maioria dos entrevistados foi bastante sucinto nas descrições dos seus procedimentos de escolha. Resumem-se, no caso do entrevistado 5, apenas à escolha entre várias *Framework*s destacando as que considerou mais importantes, e posteriormente, com ajuda de protótipos funcionais para identificar vantagens e desvantagens das diferentes ferramentas, selecionar aquela pela qual detêm de mais experiência. O entrevistado 4 defendeu que quando um projeto implica o desenvolvimento de várias aplicações com conceitos diversificados ter o mesmo processo para a seleção de ferramentas em todas as decisões pode não ser a melhor solução, e por isso deixa de existir um processo padrão entre a equipa. O critério de flexibilidade está bastante presente em projetos desta tipologia, conforme as suas funcionalidades se elege o que melhor se enquadra ao mesmo, a escolha é feita em função do que o projeto exige, necessitando desta forma de se reforçar o critério mencionado, uma vez que eventualmente em projetos futuros se poderá abandonar ferramentas que foram usadas e iniciar umas novas. Neste caso o entrevistado 4 ainda mencionou para concluir: "… temos de estar sempre em adaptação".

O entrevistado 3 descreveu um processo bastante detalhado e dividido por várias fases. Iniciou o discurso justificando que existem muitas *Framework*s no mercado que ajudam a desenvolver *single page applications*: "Haviam muitas *Framework*s no mercado que interessavam e que ajudavam a desenvolver SPA, portanto fomos analisar quais é que existiam e fomos fazer um estudo comparativo entre elas". Posto isto, relatou a fase 0 do processo de seleção de uma *Framework*. Esta fase compreendia duas etapas, a seleção das diferentes *Framework*s que poderiam ser utilizadas para aplicações do tipo SPA e a implementação de uma aplicação que serviu para um estudo comparativo entre as 9 selecionadas. A aplicação consistia num gestor de conhecimentos entre utilizadores e tinha como objetivo avaliar a potencialidade que cada *Framework* tinha. Na fase 1 foi elaborada uma tabela comparativa entre os diferentes critérios e potencialidades das ferramentas, estas foram classificadas com notas de 0 a 3, sendo 0 o mais baixo e 3 o mais elevado, onde, da soma das notas resultaram apenas quatro *Framework*s com pontuação mais elevada, e as restantes cinco foram excluídas. A fase 2 já consistiu em acrescentar novas funcionalidades à aplicação anteriormente descrita, tais como, *routing*,

tag cloud, atualizações parciais da página, entre outros. O resultado desta análise foi um gráfico dividido em "complexo/completo", e daí se concluiu que no extremo desse gráfico se encontrava a *Framework* Backbonejs que era bastante simples mais muito pouco poderosa e na outra extremidade encontrava-se o Emberjs que era das *Frameworks* mais completa mais também das mais complexas. Deste modo, achou-se que a melhor estaria no meio-termo e portanto a decisão final prendeu-se entre o Angularjs e o Knockoutjs. Por fim, foi feita uma avaliação de conceito. Desenvolveu-se uma nova aplicação para reservas de salas e com as duas restantes *Framework*s foram elaborados testes à integração destas com um servidor, ligação de dados, css, pedidos ajax, etc. Concluiu-se que ambas cumpriram as tarefas propostas e deste modo a diferença entre estas, a nível técnico, não era muito significativa. Prosseguiu-se a análise com um outro tipo de comparações e estas é que ditaram a escolha. Foi elaborada uma pesquisa a *websites* como github, stackoverflow, entre outros, para ver quantas pessoas contribuem com a *Framework*, quantas perguntas e soluções existem, quantos problemas já foram resolvidos ou estão por revolver, etc.. Perante estas comparações a escolha final foi dirigida à *Framework* Angularjs e é esta que é recomendada pelo entrevistado 3 para o desenvolvimento dos seus projetos. Pôde-se, desta forma, concluir que não existe nenhum processo *standard* que seja conduzido de igual forma pelos entrevistados, cada empresa e cada projeto exigem e possuem diferentes métodos de seleção para as decisões a serem tomadas.

Após a descrição dos processos de escolha da(s) *Framework*(s) foi pedido aos entrevistados que listassem os principais critérios utilizados para ajudar na decisão pela *Framework* final. Da análise dos dados obtidos foi possível construir a seguinte tabela na qual são apresentados os critérios mais mencionados pelos entrevistados:

Critérios	**Nº de ocorrências**	
Comunidade	3	- "Se houver uma comunidade grande consigo mais facilmente encontrar ajuda e artigos e consigo falar com outras pessoas que usam"; (E2)
Documentação	2	- "A quantidade de informação que existe na internet sobre esta FW"; (E2)
Compatibilidade	3	- "Compatibilidade com maior parte dos dispositivos passados por parte do cliente"; (E5)
Curva de aprendizagem	3	- "Até que ponto é difícil trabalhar com aquela *Framework* ou não"; (E3)

Critérios	Nº de ocorrências	
		- “Facilidade de aprendizagem dessas ferramentas mais semelhantes”; (E5)
Experiência	2	- “Já trabalhava com ela, não foi preciso passar por uma fase de aprendizagem”; (E2)
Plugins	2	- “Nem todas as funcionalidades estão presentes na *Framework*, até que ponto existem extras que consigo juntar à ferramenta para produzir as partes que lhe faltam”; (E3)
Requisitos do projeto	1	- “Olhamos para o projeto e escolhemos aquilo que ele nos pede”; (E4)
Popularidade	2	- “Se ainda está atualizado e em alta no mercado”; (E5)
Open source	1	
Atualizada	1	
Funcionalidades	1	
Tamanho	1	
Número de linhas de código	1	- “Escrever muito código é porque a *Framework* dá muito trabalho, ajuda-me pouco”; (E3)
Performance	1	
Maturidade	1	
Dependências	1	
Flexibilidade	1	

Tabela 3 - Lista dos principais critérios utilizados para ajudar na decisão pela *Framework* final

Da Tabela 3 é possível verificar que os critérios mais mencionados e considerados até mesmo prioritários pelas fontes de informação são a documentação, comunidade, compatibilidade e curva de aprendizagem.

Após a descrição do processo de escolha da *Framework* foi pedido aos entrevistados que identificassem as dificuldades que possam ter ocorrido nesse processo de seleção. Embora dois dos entrevistados não tenham identificado dificuldades, a maioria dos entrevistados referiu como dificuldades o número alargado de *Frameworks* existentes, que respondem ao mesmo problema. O que, segundo os entrevistados, torna todo o processo um pouco mais demorado e complexo, obrigando ainda a que haja uma rigorosa definição dos critérios para a seleção.

A maioria dos entrevistados quando questionados sobre a duração do processo de escolha da *Framework* mencionaram tempos bastante diversificados, variando entre 1

semana a 2 meses, dando para concluir que, embora o tempo seja essencial na seleção da *Framework* mais adequada a uma projeto, por vezes, o período de desenvolvimento de um projeto contraria bastante essa afirmação e obriga a que esse tempo seja encurtado para que os prazos sejam cumpridos.

Foi requerido aos entrevistados que partilhassem se o resultado final das suas escolhas foi unânime entre os vários elementos das suas equipas. Em geral as respostas foram positivas tendo a maioria confirmado que as decisões são concordantes. Apenas num dos casos essa decisão depende primeiro das exigências do cliente, se este já impuser alguma ferramenta não existe qualquer escolha a fazer, caso contrário a escolha da ferramenta não costuma ser universal pela equipa, defendeu o entrevistado em questão, acrescentando que depende sempre do fim que se quer atingir. Por vezes, decisões finais podem levar a alguma inconsistência entre opiniões numa equipa de trabalho, contudo, através dos entrevistados em questão, pôde-se concluir que a seleção de uma *Framework* não é um aspeto que cause incompatibilidades entre os diferentes elementos das suas equipas de trabalho.

Um projeto web ou móvel não se baseia apenas em ferramentas de javascript para o código front-end client-side, existem também outras ferramentas que podem auxiliar no código CSS ou em programação para plataformas móveis. No entanto, a utilização de diferentes ferramentas potência que hajam incompatibilidades, mais ou menos complexas, entre elas. Cada entrevistado foi questionado sobre as *Frameworks* que usavam para CSS e dispositivos móveis, o que permitiu identificar as ferramentas abaixo listadas como sendo as mais usadas:

- Bootstrap (4)
- Jquery Mobile (1)
- Aplicações nativas (1)
- Foundation (1)
- Modernizr (1)
- Less (1)

Pelos resultados anteriores foi possível concluir que a *Framework* de CSS Bootstrap foi a ferramenta mais utilizada pelos inquiridos, havendo respostas bastantes construtivas a confirmar a abordagem antecedente tais como: “começamos por utilizar a abordagem de Mobile First e para isso precisávamos de algo que facilitasse a nível de compatibilidades e outras vantagens que viemos a descobrir” [entrevistado 5] e “…bootstrap, já a pensar nos vários dispositivos e resoluções a adaptar o design”

[entrevistado 4]. Após a identificação de outras *Frameworks* utilizadas era importante ficar a conhecer a existência de problemas que possam ter ocorrido aquando a junção das várias ferramentas. Nenhum dos entrevistados levantou qualquer inconveniente para este aspeto, acrescentando até que a minoria não compensa o esforço de alterar alguma decisão. Deste modo, concluiu-se que pode não existir impossibilidade de utilizar várias ferramentas para objetivos diferentes, para o mesmo código front-end de um projeto client-side.

Foi requerido aos entrevistados que descrevessem a ordem de decisões no front-end no início dos seus projetos, as várias fases por que passaram até começarem a desenvolver e a programar. Ao longo da descrição do processo de escolha da *Framework*, nos parágrafos anteriores, foi possível concluir, como previamente mencionado, que não existe uma norma que permita assumir que o processo deva seguir determinados procedimentos. Da mesma forma se pôde concluir que a ordem de decisões é um aspeto, igualmente, não universal, cabendo à equipa a condução dos mecanismos para o processo de escolha de uma *Framework*.

Da análise do critério de decisão curva de aprendizagem, tendo sido este um dos critérios mais referidos (Tabela 3) verificou-se que a maioria dos entrevistados referiu que essa curva pode ser influenciada pelo tempo de desenvolvimento do projeto e pelas necessidades inerentes ao projeto. Alguns dos entrevistados referiram ainda que o tempo médio de aprendizagem para os programadores é de 2 meses, com existência de formações em alguns casos. Apenas um entrevistado referiu um tempo de aprendizagem menor, devido à tipologia dos projetos e do reduzido tempo despendido no desenvolvimento. Pelos resultados obtidos, foi possível concluir que a curva de aprendizagem pode ser obrigada a encurtar de acordo com o tempo que é oferecido para o domínio da ferramenta, tempo esse que é influenciado pelas necessidades do projeto e pelo tempo disponível para o seu desenvolvimento.

Por fim, os entrevistados foram questionados acerca da sua satisfação para com a *Framework*, e foram obtidas algumas conclusões interessantes, tais como:

- “Sim. Até agora não tenho razão de queixa.” – AngularJs
- “Estou satisfeito com a compatibilidade entre browsers (grande maioria), os exemplos e dicas abundantes para soluções, a quantidade de plugins e a qualidade da ferramenta.” - Jquery
- “Jquery é bom pelos plugins, como validações que usamos muito no dia-a-ia. Pessoalmente gosto e não gosto. O poder das *Framework*s facilita bastante a implementação, como validações, mas depois também tenho a impressão que dá

muito mais trabalho se quiser personalizar em termos de design. Tendo em conta os projetos que temos muito distintos, temos sempre trabalhos extra de personalizar os formulários, as vezes até é capaz de ser mais rápido fazer manualmente do que implementar algum plugin. Já houve casos que me arrependi da *Framework*, e também já houve casos de incompatibilidades como por exemplo a API do fb porque eles estavam sempre a mudar a API deles."

Em suma, todos os entrevistados se mostraram, pessoalmente, satisfeitos com as suas decisões. Mesmo que essa satisfação não coincida com a decisão final da *Framework* para o projeto, todos eles demonstraram positivamente que a *Framework* escolhida por eles foi a melhor opção e estão satisfeitos com os resultados obtidos após e antes da decisão final. Um resumo da análise descrita anteriormente pode ser visualizado na tabela do Anexo 2 – Resumo da análise dos dados da primeira entrevista.

V. PROJETO EM CONTEXTO EMPRESARIAL

A. Descrição da empresa e do projeto

A empresa prodcent, LDA é uma firma de desenvolvimento de plataformas comerciais para empresas a nível mundial. Uma forma de alojar, tratar, promover e disponibilizar informação empresarial de produtos, através da internet para o Mundo.

Esta empresa encontra-se a aproximadamente um ano e meio de execução, sendo o Engenheiro Pedro Sousa Rêgo o responsável e CEO por todo o portal em desenvolvimento. Um dos projetos que esta empresa se encontra a desenvolver atualmente é um portal web e móvel de produtos multilingue. O projeto é denominado de centroproduto, e neste momento é o trabalho principal a decorrer nesta entidade.

O centroproduto está inteiramente ligado a produtos e empresas e na divulgação dos seus catálogos e perfis. Trata-se de um portal para alojar todo o tipo de produtos distribuídos por várias categorias e empresas. Uma das suas melhores características é o facto de permitir conteúdo multilingue possibilitando, desta forma, alargar as suas funcionalidades a nível mundial, assim como, divulgar empresas e produtos internacionalmente.

Este projeto entrou em funcionamento experimental em contexto real no dia 24 de Fevereiro, tendo iniciado o alargamento da sua base de clientes a partir de 15 de Junho. É um projeto que nunca se encontrará finalizado, já que o desenvolvimento contínuo faz parte do seu conceito, pelo que todas as características mencionadas neste documento farão referência ao estado atual do projeto.

O centroproduto disponibiliza às empresas registadas funcionalidades para inserção de dados em ficha de empresa, registo ou importação de dados de produtos para fichas de produto normalizadas e classificadas, e configuração flexível de catálogo de produtos. O conteúdo inserido tem suporte multilingue, do portal para sete línguas, e dos conteúdos para mais de 50 línguas. Permite multiutilizador por empresa e multiempresa por utilizador prevendo os casos de utilizadores de grandes empresas e grupos empresariais. Disponibiliza ferramentas avançadas de importação, configuração e gestão fácil dos catálogos com centenas ou milhares de produtos, ao mesmo tempo que faculta mecanismos simples de criação e edição de produtos para pequenas alterações ou empresas de muito pequena dimensão.

Para os visitantes sem sessão iniciada o *website* oferece um sistema de montra de produtos com navegação, e seleção por imagens, diretórios por categorias de produto e ferramentas de pesquisa dentro de conteúdos, incluindo pesquisa por produto e por

empresa, sempre com capacidade multilíngue. O projeto acolhe ainda uma camada de funcionalidades destinadas à gestão própria do projeto sobre o portal, incluindo a criação e gestão de estado dos utilizadores, autorização de empresas candidatas, validação das imagens publicadas quanto à natureza e qualidade, e outras.

O portal centroproduto, como mencionado no capítulo inicial de introdução a esta dissertação, tem como público-alvo o mercado B2B e encontra-se apresentado em dois *websites* paralelos, ambos servidos por uma mesma API, sendo um adaptado para versão desktop e outro para dispositivos móveis. Estes dois sítios web têm muito de comum a nível de interfaces e funcionalidades, contudo têm algumas diferenças significativas que se justificam por objetivos algo distintos. Enquanto o sítio web móvel dá suporte a todas as tarefas que tipicamente possam ser requeridas quando em deslocação ou lazer, o sitio web adaptado ao desktop disponibiliza adicionalmente pacotes de funcionalidades de gestão de catálogo e do portal que requerem um ecrã de dimensão superior, teclado e rato ou maior capacidade de processamento.

Esta empresa ultrapassa os 10 elementos atualmente, abrangendo as áreas técnicas de sistemas, usabilidade e design web, bases de dados e programação, testes de qualidade, linguística, marketing e ação comercial, além de gestão operacional e desenvolvimento do negócio. Quando se iniciou o processo de investigação na empresa a separação entre front-end e back-end era bastante clara, havendo três elementos responsáveis pela lógica e modelo dos dados e outros três responsáveis pelo design, estrutura e funcionalidades da plataforma. A comunicação que iria servir de ponte a estas duas áreas também já havia sido estipulada antes do início da investigação, seria uma API Json que transportaria os dados, de uma base de dados, de forma estruturada, para serem tratados pela *Framework* javascript a ser escolhida posteriormente. Também, grande parte da lógica de tratamento e modelo dos dados já estava concluída, faltando, assim, na altura do início da investigação ilustrar e processar esse modelo do lado do front-end com um design e uma estrutura mais adequada à atual. Entre os três elementos de front-end existia uma separação, um designer que tratava toda a personalização do website, já programando CSS e HTML estático, e dois programadores de javascript e HTML que após o design concluído tratariam do dinamismo das páginas estáticas criadas.

Quando se iniciou o processo de investigação na empresa, havia como principais tarefas todo o procedimento de escolha de qual a *Framework* javascript a utilizar para se poder iniciar o código Front-end do website. Para além da decisão pela *Framework* era necessário ter em atenção vários aspetos, tais como compatibilidades, outras

Frameworks de CSS e móveis que se poderiam vir a utilizar, *Search Engine Optimization (SEO)*, *client-side* ou *server-side*, entre outros aspetos que serão mencionados posteriormente. Após a decisão final era necessário estudar o funcionamento da *Framework* e começar a programar o *website*.

A criação de produtos, catálogos e empresas, a manutenção de ficheiros e produtos para associar às criações anteriormente citadas, a atualização ou remoção frequente de conteúdo, entre outros, são interações elaboradas frequentemente pelos utilizadores no *website* e necessitam de ser interpretadas em tempo real, cada modificação, cada criação ou cada remoção. Desta forma, o conceito de CRUD (*Create, Read, Update, Delete*) é um tópico com bastante efeito nesse tipo de interfaces.

No início do desenvolvimento deste projeto, e após definidas as funcionalidades principais afetas ao mesmo, foi indispensável escolher as ferramentas Javascript para a programação front-end que respondessem adequadamente aos requisitos e à complexidade do website. Um dos motivos principais pela decisão de ser utilizada uma *Framework* javascript deveu-se especialmente à evolução recente do AJAX e pela quantidade de código corrido pelo cliente ser cada vez mais e com maiores capacidades. O facto de ser necessária a atualização apenas de pequenas porções da página e não propriamente da página completa também contribuiu para essa decisão de se utilizar uma *Framework* de javascript. O efeito CRUD referido anteriormente também reforça a declaração anterior. Se por cada atualização, remoção, ou criação de conteúdo, se estivesse constantemente a atualizar a página completa do *browser*, tornar-se-ia aborrecido para o utilizador, assim como a performance do website seria afetada. Por último e não menos importante, como se trata de um portal com grandes funcionalidades a nível de carregamentos de dados e quantidade de informação disponibilizada, a organização do código tornou-se um aspeto indispensável, acrescentando-se também, a necessidade de separação das suas partes lógicas e reutilização dessas mesmas componentes.

Quando se iniciou a investigação, já haviam sido estipuladas, pelo gestor, duas *Frameworks* para estudar e analisar. Embora a investigação tivesse abrangido mais algumas, a decisão partiu sobretudo pela *Framework* Angularjs e pela Biblioteca BackboneJs. No final decidiu-se pelo AngularJs, tendo, esta decisão, sido sujeita a posterior aprovação e análise do gestor de projeto.

B. Processo de escolha da *Framework* javascript na empresa

No início da investigação na empresa, já tinha sido estabelecido que a *Framework* escolhida teria de se basear no modelo MVC para melhor organização e reutilização de todo o código front-end. As *Frameworks* Angularjs e BackboneJs já estavam pré-selecionadas pelo gestor, para futura análise e comparação. Os motivos desta pré-escolha deveram-se essencialmente à popularidade e maturidade destas duas *Frameworks*. Também o facto de a *Framework* Angularjs ser desenvolvida pela empresa Google e a maturidade da *Framework* BackboneJs, tornaram-se aspetos muito relevantes nesta seleção inicial.

Durante todo o processo de escolha, que será detalhado nos parágrafos seguintes, não houveram grandes dificuldades sentidas, e, uma vez que a opção se baseou apenas nas duas *Frameworks* referidas tornou a decisão bastante mais simples, rápida e acertada. Decorreram três semanas até à deliberação final e essa resolução foi unanime entre equipa. A análise e a escolha foram elaboradas por um programador *front-end* e pelo gestor de projeto e posteriormente consentidas pelo gestor.

Foram estudadas sete *Frameworks*, entre as quais, jquery, angularjs, knockoutjs, emberjs, backbonejs, javascriptmvc e canjs. Estas *Frameworks* foram selecionadas para análise devido a uma pesquisa detalhada por vários websites[57], que foram filtrados de acordo com a sua atualidade e popularidade a nível de respostas e comentários favoráveis ao mesmo; livros e artigos a mencionar *Frameworks* de javascript, onde se pôde concluir que estas sete referidas, pela frequência e ordem pela qual surgiram nos resultados da recolha de informação, seriam as ideais para analisar e comparar. Na Ilustração 5 é possível verificar o grau de importância e interesse que algumas das *Frameworks* mencionadas têm vindo a ganhar ao longo do tempo. Contudo, devido à urgência de iniciação do desenvolvimento front-end do projeto, e uma vez que, como referido anteriormente, a maioria da lógica de tratamento e modelo dos dados já estava terminada, a análise das *Frameworks* selecionadas teve de se resumir às duas citadas no início deste tópico, AngularJs e BackboneJs, também pelos motivos supracitados na distinção destas duas *Frameworks*.

[57] Exemplos de websites: http://codebrief.com/2012/01/the-top-10-javascript-mvc-*Framework*s-reviewed/ - 23/09/2013; http://todomvc.com/ - 23/09/2013;

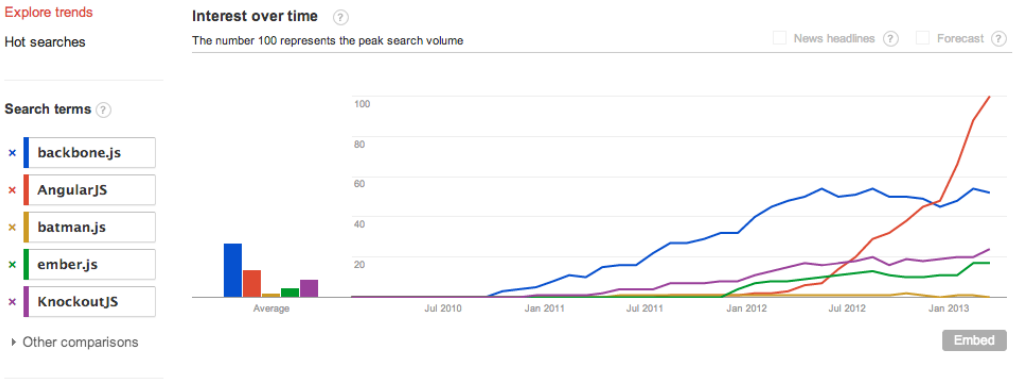


Ilustração 5 - Trending de algumas *Frameworks* JavaScript MVC na Google

Seguidamente foi elaborado um estudo preliminar acerca da definição do modelo MVC e a caracterização das suas diferentes componentes: *controllers*, *views* e *models*. Como revisto anteriormente no capítulo das *Frameworks* MVC de javascript, nem todas seguem este modelo em particular, contudo, procuram seguir toda a sua estrutura lógica ao conceder organização e definição das diferentes partes lógicas da programação. A Tabela 4 ilustra, de entre as *Frameworks* estudadas, o detalhe e rigor do modelo MVC que estas propõem:

Framework	**Modelo**
Jquery	Não segue o modelo MVC
AngularJs	MVW (Model View Whatever)
BackboneJs	MVP (Model View Presenter)
CanJs	MVC (Model View Controller)
EmberJs	MVC (Model View Controller)
JavascriptMVC	MVC (Model View Controller)
KnockoutJs	MVVM (Model View View Model)

Tabela 4 - *Frameworks* e modelo mvc

Durante o processo de investigação das *Frameworks*, surgiu a dúvida acerca de qual o lado pela qual deveria ser desenvolvido o website, se pelo lado do cliente ou do servidor. Esta hesitação surgiu devido ao client-side ainda ser, atualmente, um tabu para muitos programadores, refletindo-se este aspeto pela maioria dos *websites* de grande prestígio, tais como, linkedin.com, facebook.com, alibaba.com, twitter.com, ainda serem programados pelo lado do servidor. Apesar dos aspetos citados serem relevantes, o client-side encontra-se, atualmente, em ascensão muito devido à evolução do conceito de Ajax, baseado em javascript, que se trata de uma técnica para criar páginas web de

forma dinâmica e com atualizações assíncronas, permitindo trocas de pequenas quantidades de informação com um servidor e, desta forma, evitar o recarregamento do DOM por inteiro ("AJAX Introdução," s.d.). Existe, também, cada vez mais código javascript a correr nos *browsers*, uma vez que estes começam a ter mais capacidades de processamento e memória, diminuindo, deste modo, a latência do website e a oportunidade de se ter uma *Single Page Application*. Estes aspetos referidos acerca da importância que o client-side tem vindo a ganhar superaram qualquer tabu mencionado previamente, prosseguindo-se, assim, com a ideologia inicial de desenvolver uma plataforma client-side.

Considerou-se a maturidade e a popularidade como sendo os dois principais critérios utilizados na seleção da *Framework*. Contudo, apenas estes dois não seriam suficientes para uma comparação detalhada e exigente de qual a *Framework* mais adequada ao *website* a desenvolver. Foi, deste modo, elaborada uma pesquisa acerca de quais os critérios essenciais na distinção de *Framework*s web, dos quais se destacaram a curva de aprendizagem, a documentação, performance, tamanho, dependências, compatibilidade, open source, comunidade e acessibilidade/facilidade de aprendizagem (Graziotin & Abrahamsson, 2013).

Definidos os critérios, era necessário iniciar a análise comparativa entre as duas *Frameworks* selecionadas, tendo sempre por base as necessidades funcionais do projeto e a experiência do programador. Inicialmente foi elaborado um estudo breve acerca das duas *Framework*s; quais os convenientes e inconvenientes das mesmas, os projetos já implementados, fóruns e reviews com comentários e críticas favoráveis ou desfavoráveis às *Framework*s, e por último, a comunidade online de suporte, como o Github, Stackoverflow, revistas de informática, entre outras. Desta primeira análise foi possível concluir que ambas as *Framework*s são bastante poderosas tanto a nível de comunidade, como funcionalidades disponibilizadas ao programador. Embora BackboneJs seja uma *Framework* que "oferece" mais liberdade de programação ao programador, a *Framework* AngularJs apresenta um modelo de binds[58] em tempo real, assim com todo um sistema de modelos de HTML incorporados (Runeberg, 2013). Também foi possível observar que BackboneJs tem a vantagem de ser mais compacto a nível de peso e tamanho, contudo engloba mais dependências (jQuery ou Zetpo, underscore.js, entre outras) que a *Framework* AngularJs. Também a nível de performance foi possível verificar que

[58] "*Data-binding in Angular apps is the automatic synchronization of data between the model and view components. The way that Angular implements data-binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. When the model changes, the view reflects the change, and vice versa.*" - https://docs.angularjs.org/guide/databinding (13/03/2014)

Backbone por ser mais simples implica menos problemas a este nível, contudo não garante que Angularjs seja problemática nesta vertente se for programada segundo as suas boas práticas. Consequentemente a esta primeira abordagem pelas duas *Frameworks*, que permitiu retirar já algumas noções, foi elaborada uma pesquisa por vários artigos de acordo com cada um dos critérios previamente mencionados e conseguiram-se retirar os resultados que podem ser visualizados na tabela seguinte (Tabela 5) ("Backbonejs vs Angularjs : Demystifying the myths," 2013; Graziotin et al., 2013; Addy Osmani, 2012a; Runeberg, 2013):

Critérios	Angularjs	BackboneJs
Maturidade	-1	+1
Popularidade	+1	+1
Curva de aprendizagem	+1	-1
Documentação	+1	-1
Performance	+1	+1
Tamanho	-1	+1
Dependências	+1	-1
Comunidade	+1	+1
Facilidade de aprendizagem	-1	+1
Open source	+1	+1

Tabela 5 - Resultados dos critérios analisados

Após uma comparação mais detalhada das duas *Framework*s com base nos critérios mencionados anteriormente, ainda não era possível retirar uma conclusão final para a escolha de qual a *Framework* mais adequada ao projeto. Desta forma, foram utilizados outros métodos para auxiliar no processo de decisão. Começou-se por estudar e desenvolver alguns exercícios com ambas as *Frameworks* para ver como funcionavam e conhecer o seu grau de dificuldade. Para o desenvolvimento destes exercícios foi utilizado o website todomvc que oferecia uma miniaplicação de exemplo, programada para cada uma das *Frameworks* mvc disponíveis no website e, desta forma, foi possível comparar as diferentes *Frameworks*[59] ao nível de desempenho, linhas de código, sintaxe e facilidade de aprendizagem. Como resultado desta comparação foi possível verificar a existência de algumas diferenças entre as *Frameworks*. Pela experiência do programador era importante que a *Framework* de javascript não excluísse a criação do *design* e da

[59] Backbone.js, AngularJs, EmberJs, KnockoutJs, Dojo, Yui, Agility.js, Knockback.js, CanJs, Maria, Polymer, React, (…)

estrutura do website pela linguagem HTML, por forma a conciliar, também o código entre a equipa de front-end. Este aspeto retrata uma das maiores diferenças entre as duas *Frameworks*, enquanto BackboneJs está mais direcionado para código em javascript, sendo os modelos HTML desenvolvidos dinamicamente em javascript também, a *Framework* Angularjs disponibiliza os seus próprios modelos e diretivas[60] de HTML permitindo uma maior interação com a estrutura do *website* e, deste modo, facilitar a aprendizagem do programador.

Outro fator importante que contribuiu para a decisão final foi a facilidade de aprendizagem. Foram investigados tutoriais, vídeos, entre outros meios que facilitassem a rápida aprendizagem das duas *Frameworks*. Optou-se por iniciar o estudo das *Frameworks* pela Angularjs e através de alguns vídeos tutoriais verificou-se que a rapidez com que se aprendeu o funcionamento e sintaxe da *Framework* foi satisfatório, assim como a facilidade de aprendizagem do programador de javascript e do *designer*, permitindo também, desta forma, ambos poderem trabalhar quase individualmente, necessitando apenas de unir os códigos de javascript e HTML com as funcionalidades e diretivas da *Framework* AngularJs. Como estava para breve o início do desenvolvimento do projeto optou-se por escolher imediatamente a AngularJS pelos motivos supracitados e também pelo motivo de esta *Framework* ser desenvolvida por alguns programadores da Google e, embora, ainda seja recente, aparentava e aparenta vir a ter grande potencial a nível de animações e funcionalidades para dispositivos móveis. Mais estudos foram feitos para comprovar que Angular seria mesmo a escolha ideal e a equipa está satisfeita com a decisão tomada.

A empresa centroproduto adotou uma abordagem de "*Mobile First*" e por este motivo era importante ter noção do que implicaria a utilização de Angularjs para programação em dispositivos móveis e se se justificaria o seu uso. Inicialmente, foi elaborado um estudo sobre que tipo de aplicação para dispositivos móveis seria, se do tipo web, hibrido ou nativo, contudo, por questões de tempo e curva de aprendizagem optou-se por uma aplicação web. De seguida foram selecionadas algumas *Framework*s para plataformas móveis e estas foram comparadas entre si segundo alguns critérios de performance, compatibilidade, adaptabilidade, responsividade, entre outros, ficando a decisão final por jquery mobile e Sencha Touch e posteriormente apenas por jquery mobile por questões de suporte, compatibilidade e dos requisitos do projeto, uma vez que

[60] "*At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's* **HTML *compiler*** *($compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children.*" - https://docs.angularjs.org/guide/directive (13/03/2014)

Sencha é uma *Framework* mais direcionada a jogos e código javascript mais complexo. Jquery mobile também aparentava ser mais percetivo a ativo na comunidade. Por último era necessário testar a compatibilidade de Angularjs com jquery mobile e justificar o uso destas duas e qual a mais adequada. Após várias investigações chegou-se à conclusão que, ou se deveria usar uma ou outra, ambas preenchiam os requisitos essenciais para dispositivos móveis e juntas só tornariam o *website* mais pesado. Desta forma, decidiu-se pela conservação do Angularjs, uma vez que o código iria ser, posteriormente, reaproveitado para a versão desktop do *website*.

Durante o processo de decisão do Angularjs optou-se também por analisar uma *Framework* denominada Foundation, para facilitar toda a estruturação do código CSS. A decisão por esta ferramenta foi tomada antes do *designer* integrar a empresa, e por este motivo interessava apenas conhecer a compatibilidade com a *Framework* Angularjs. Os testes elaborados com as duas *Frameworks* não foram muito satisfatórios havendo alguns conflitos com os ficheiros javascript de ambas as ferramentas e também o tempo e a experiência do programador não facilitaram a que houvessem análises e testes mais aprofundados e desta forma, optou-se à partida por não utilizar a *Framework* Foundation e permanecer apenas com a *Framework* Angularjs.

A escolha da *Framework* javascript foi o último processo a ser decidido pela equipa antes do início de desenvolvimento do projeto, que, após mais ou menos duas ou três semanas de estudo intensivo da *Framework*, começou finalmente a ser programado.

1. Experiência e satisfação com a *Framework* escolhida

Uma das principais conclusões que se extraiu ao se utilizar a *Framework* Angularjs na empresa foi que esta *Framework* é ideal para *websites* dinâmicos; a atualização em tempo real do conteúdo, toda a interatividade que é adicionada ao HTML e a responsividade com que a aplicação trabalha permite uma experiência muito mais "user-friendly" inibindo contantes atualizações da aplicação inteira. Pode-se concluir por estas e mais razões que serão descritas posteriormente, que a empresa se encontra bastante satisfeita com a utilização da *Framework* Angularjs na programação do código front-end do seu projeto centroproduto.

Angularjs é uma *Framework* conhecida pelas suas particularidades que a tornam muito poderosa(Schimtz & Lira, 2014). Esta funciona como uma extensão ao DOM, acrescentando novos parâmetros e interagindo de forma dinâmica com os vários elementos da página, ou seja, com a *Framework* Angularjs é possível adicionar novos

atributos no código HTML para se obter novas funcionalidades extras sem necessidade de programar essas implementações inteiramente com javascript. Esses novos parâmetros que alteram o comportamento padrão do HTML são chamados de diretivas e podem vir já de raiz com a própria *Framework* (exemplo: *ng-view, ng-click, ng-show, ng-init, ng-if, ng-app, ng-controller, ng-disabled,* etc…), como serem personalizadas em javascript e posteriormente reutilizadas ao longo do código HTML.

Outra particularidade que agradou os programadores da empresa foi o conhecido termo "*two way data binding*", que funciona como forma de ligar automaticamente uma variável ou objeto do código *Javascript* a algum elemento do HTML. Quando é feita alguma alteração nessa variável no DOM, o seu valor na programação javascript irá ser imediatamente atualizado. No Anexo 3 – One way data binding e two way data binding, é possível analisar todo este processo em mais detalhe.

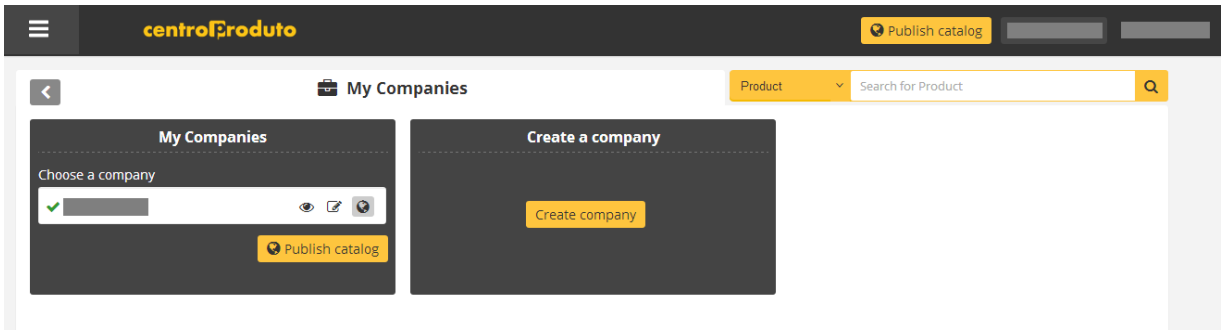


Ilustração 6 - Processo de publicação de um catálogo - catálogo não publicado

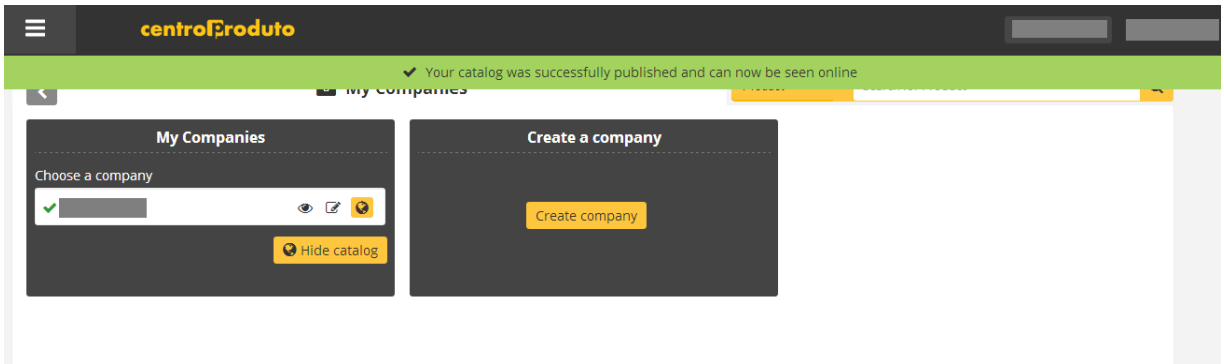


Ilustração 7 - Processo de publicação de um catálogo - catálogo publicado

Como exemplo do projeto a ser desenvolvido é possível visualizar um dos processos de publicação de um catálogo de uma entidade nas figuras representadas anteriormente (Ilustração 6 e Ilustração 7). Na Ilustração 6 o catálogo ainda não se encontra publicado pelo que se pode visualizar dois botões para realizar a ação de

publicar e ainda um ícone com uma cor acinzentada ilustrando que o catálogo ainda não está visível. Após desencadear a ação de publicar o catálogo é possível observar na Ilustração 7 que um dos botões que estaria presente em todas as páginas já não se encontra visível e o outro presente na página reflete a ação contrária à publicação. O ícone que se encontrava com uma cor pouco chamativa está agora com mais visibilidade informando o utilizador que o seu catálogo está visível mundialmente. A consequência da ação realizada também pode ser visualizada numa mensagem de sucesso. Todos estes passos foram realizados graças ao "data binding" do Angularjs que com apenas uma variável em estado "*true*" ou "*false*" pode alterar a visualização de uma página apenas de uma só vez aquando a alteração dessa mesma variável.

Uma outra qualidade que se conseguiu retirar desta *Framework* foi o conceito de "*dependency injection*". Este termo determina como um objeto obtém as suas dependências e deste modo, é possível que quaisquer outros tipos de objetos presentes na aplicação sejam injetados num controlador, diminuindo por fim a complexidade do código e favorecendo a reutilização. No exemplo anterior referiu-se o processo de publicação do catálogo. Nesse caso, esse processo pode ser efetuado aquando a visualização do catálogo, ou ao terminar sessão, ou apenas carregando no botão presente no cabeçalho na página, entre outros. Deste modo, estar a criar essa função em todos os controladores separados por páginas seria algo difícil e confuso de manter, logo, criando apenas uma função e seguidamente injetar esse serviço em cada um dos controladores que necessitem da mesma, será muito mais rentável e organizado.

Para um novo utilizador no *website*, este terá uma grande quantidade de dados a preencher, desde criação de uma empresa, até à obtenção de uma catálogo e finalmente gerar os produtos para esse catálogo. Todas estas criações de conteúdo passam por formulários com campos respetivos para o utilizador preencher de forma correta. Por forma a validar o conteúdo inserido, Angularjs mostrou-se bastante prestável neste aspeto, fornece todo um sistema de validação de campos, desde obrigatórios, máximo e mínimo de caracteres, padrões específicos de caracteres (por exemplo, só números para um determinado campo), validação de campos do tipo email, desabilitar um botão enquanto o formulário estiver inválido, entre outros. Todas estas validações são efetuadas diretamente a partir de diretivas injetadas no código HTML. Desta forma, torna-se muito mais simples de validar formulários corretamente impedindo que o utilizador insira dados demasiado incorretos, ou cometa erros sem se aperceber, dando-lhe informações válidas sobre os erros que está a cometer. A Ilustração 8 mostra um exemplo de validação de formulário de um campo de e-mail do projeto.

Ilustração 8 - Validações de formulário - AngularJs

Outro fator relevante que contribuiu para a satisfação com o Angularjs é toda a sua modularidade e sistema de modelos de HTML. Esta *Framework* permite não só o dinamismo no DOM ao se inserir variáveis dinâmicas entre os seus elementos, como também oferece uma diretiva de "ng-include" que disponibiliza a criação de uma página completamente separada e com o seu próprio controlador que pode ser partilhado ao longo de toda a aplicação. Como exemplo do *website* centroproduto, tem-se toda a lógica da caixa de pesquisa e de menu principal distribuídos por um *ng-include* que pode ser chamado em qualquer parte do código HTML. Outro método de criação de modelos de HTML é a conceção de diretivas personalizadas. Por exemplo, o programador pode querer fazer uma *dropdown* personalizada sem necessidade de utilizar a que é disponibilizada pelo *browser*. Desta forma, este, simplesmente cria o HTML devido para essa *dropdown* com os seus estilos de css, e injeta esse template numa diretiva personalizada. Assim, em qualquer parte do código HTML, neste caso específico, sempre que se necessite de *dropdowns* no *website* basta colocar o nome dado à diretiva no meio do código e o Angularjs responsabiliza-se pelo resto, mostrando a pequena porção de código criado para a *dropdown* no sítio respetivo do código. Desta forma é possível reutilizar várias porções de código HTML por forma a seguir padrões de *design* e estruturação do website.

Todas as diretivas do angularjs são consideradas como classes quando o código HTML é processado. Desta forma torna-se simples criar padrões de *design* para determinados casos e classes no HTML. No exemplo anterior, na Ilustração 8, acerca de validação de formulários a caixa de texto aparecia com uma borda vermelha significando que algo incorreto prevalecia naquele campo. Para esse caso, o HTML processado pelo Angularjs estaria com uma classe "*ng-invalid*" nesse campo enquanto o utilizador não digitasse o conteúdo correto, sendo essa classe substituída posteriormente por "*ng-valid*". Desta forma, basta no css definir o estilo para "*ng-invalid*" e sempre que qualquer campo esteja inválido o estilo definido irá ser visualizado. Este método torna-se bastante

vantajoso para padronizar os estilos de determinados casos no site, seja campos inválidos, seja *mouseover*, seja cliques, e principalmente animações que posteriormente virão a ser utilizadas, pelo menos, no *website* para dispositivos móveis.

Como forma de criar comunicação entre o back-end e front-end, na empresa, utiliza-se uma API Json. Angularjs fornece bons procedimentos de conciliar esta comunicação através da sua propriedade "$*http*" que permite utilizar os métodos essenciais para se cumprir o processo de CRUD que é tão convenientemente efetuado pelo Angularjs. Métodos como *".get, .post, .put, .delete, …*" tornam todo o processo de requisição Ajax muito mais simples para obter dados vindos de um servidor e tornar o website assíncrono.

O *website* centroproduto tem como uma das suas grandes finalidades a divulgação de produtos. Esses produtos podem ser visualizados através de catálogo, da pesquisa ou de categorias. O utilizador ao navegar por uma das funcionalidades descritas anteriormente irá deparar-se continuamente com uma lista de produtos respetivamente paginados; essa lista pode variar entre um a milhares de produtos sendo necessário saber gerir estas listas a nível de performance do *website*. A *Framework* AngularJs fornece uma propriedade denominada "*ng-repeat*" para o efeito permitindo criar um *loop* de uma variável que é geralmente um *array* de dados, contudo, esta prática torna-se um pouco insustentável para grandes listas e funcionalidades no topo das mesmas. Esta *Framework* é valorizada pelos seus "$*watchers*" a tudo o que é dinâmico no *website* para conseguir atualizar e despoletar qualquer alteração ao DOM imediatamente. Esta ação de observar listas de milhares de produtos pode ser problemática e até já demonstrou algumas dificuldades a nível de performance e memória no *website* centroproduto. Tem-se como exemplo um gestor de produtos implementado no *website*; esta página permite ao utilizador gerir os seus produtos e associá-los a catálogos ou categorias. Como gestão de produtos entende-se por filtragem por certos conteúdos, ordenação por determinados campos, pesquisa por certos assuntos, etc. Todas estas funcionalidades a serem "visualizadas" por cada interação na página em cima de milhares de produtos podem ser realmente penosas como já foi possível verificar aquando a inserção de grandes quantidades de dados por parte de uma empresa. A página apresentava-se seriamente lenta e cada clique para seleção de um produto demorava certa de um ou dois segundos a reagir, tal como a ordenação ou filtragem desses produtos que durava cerca de meio segundo a processar. Estes acontecimentos para o utilizador são suficientes para fazê-lo desistir da aplicação e levantar uma má critica acerca da mesma. No entanto, existem outras soluções que podem facilitar estes casos tais como *srolls* infinitos, limitação do

número de conteúdo a ser mostrado por paginação ou botões, criação de métodos que são ininterruptamente averiguados quando o utilizador realiza alguma ação, como por exemplo os filtros, em vez de permitir ao Angularjs observar cada interação na página.

Uma outra desvantagem que a Angularjs poderá demonstrar, tal como outras *Frameworks* desta categoria possam vir a manifestar, está relacionada com a memória do *browser*. Funcionalidades do lado do *browser* podem ser realmente uteis e amigáveis para o utilizador dando prestigio a um website, contudo, a memória do browser e do computador vai ficando afetada com grandes quantidades de processamento. Para computadores mais débeis e *browsers* mais antigos este assunto pode ser problemático. O ideal será, talvez, saber conciliar as funcionalidades necessárias ao client-side e as restantes para o server-side.

Por último e mais recente, o facto de se estar a produzir um *website* bastante dinâmico e SPA pode conduzir a alguns procedimentos extra para permitir a interação com as redes sociais como facebook, linkedin, ou com os motores de busca. Uma *Single Page Application* servida de Ajax é vista apenas como sendo uma única página com URLs dinâmicos que mostram diferentes conteúdos conforme o endereço, contudo, para as redes sociais ou os motores de busca essa aplicação é apenas categorizada como sendo uma única página e não têm capacidade de gerar os URLs para observar o resto do conteúdo. Deste modo, é preciso utilizar métodos extras para simular essas páginas estáticas separadamente e servi-las aos motores de busca. Com Angularjs tem vindo a ser complicado ainda utilizar esse método. Iniciou-se pela criação de *snapshots* que basicamente tinham o conteúdo das páginas estáticas e já processadas, como se fossem páginas separadas servidas por um servidor. Seguidamente foi elaborada uma configuração no servidor para existir redireccionamento com destino a esses ficheiros quando um pedido em formato *"?_escaped_fragment_=*" que é a configuração entendida pelas outras redes como sendo páginas servidas por Ajax. Contudo, este método ainda não pareceu suficiente dada a quantidade de produtos que cresce exponencialmente no *website*, logo, iria gerar demasiadas páginas estáticas, e consequentemente sobrecarregaria o servidor.

VI. AVALIAÇÃO DAS *FRAMEWORK*S MENCIONADAS

Pelos resultados obtidos anteriormente na análise das entrevistas e em todo o processo de escolha da *Framework* Angularjs na empresa pôde-se concluir que a tipologia do projeto é o aspeto primordial para definir que requisitos e critérios devem ser avaliados na escolha de uma *Framework* javascript. Desta forma, só após definida a tipologia do projeto e quais os seus principais requisitos se deve começar a pesquisar *Frameworks* que correspondam às necessidades de um projeto. A *Framework* em estudo até pode transparecer ser muito popular e ser realmente eficaz nas suas funcionalidades, contudo, pode simplesmente não se enquadrar para as necessidades de um projeto. Tem-se como exemplo o projeto da empresa, que se trata de um portal para comunicação e carregamento de dados entre empresas, e, para esta tipologia, uma *Framework* direcionada para animações e efeitos, por muito eficaz que se apresentasse, não se enquadrava neste tipo de projetos.

> "*Picking a Javascript Framework isn't about preference. It's about what best fits your project.*" – (Orsini, 2014)

Foi possível concluir, também pelos resultados obtidos previamente, que a evolução do Ajax/javascript foi o motivo principal pela opção de se utilizar uma *Framework* de javascript no código front-end de um projeto. Outros aspetos, não menos importantes, foram a necessidade de organização e reutilização do código e a experiência do programador que irá desenvolver o programa. De uma forma geral, foi também conclusivo, pelos resultados obtidos, que este tipo de decisões foi maioritariamente elaborado pela equipa de desenvolvimento e posteriormente decidido pelo gestor do projeto.

Todos os processos de decisão de uma *Framework* descritos se mostraram idênticos, tendo sempre como variante, a tipologia do projeto. Ambos procuraram vantagens e desvantagens das *Frameworks* em análise, elaboraram quadros comparativos entre as mesmas segundo os critérios definidos por cada projeto, avaliaram cada uma segundo esses critérios e desenvolveram exercícios ou protótipos com as mesmas, até, finalmente, chegarem a um consentimento final. Deste modo, o único obstáculo que se conseguiu obter pelos resultados adquiridos foi o número alargado de *Frameworks* existentes atualmente que atrasavam o início do processo de decisão, uma vez que, se tornaria difícil escolher algumas *Frameworks* para analisar de entre tantas que resultariam das pesquisas elaboradas pelas equipas.

A duração desde o início do processo de escolha até à decisão final foi um aspeto muito variante entre todos os resultados obtidos. Foi um facto que dependia frequentemente da urgência de iniciação do projeto, e foi este, o principal motivo relatado durante as entrevistas e também durante o processo de escolha na empresa, uma vez que de sete *Frameworks* para se analisar se reduziu para duas apenas, devido à necessidade de se começar a desenvolver o código front-end dos *websites* o mais rápido possível. Finalizado o processo de escolha, pôde-se concluir que em todos os casos analisados a decisão final foi unanime entre a equipa de desenvolvimento e os gestores dos projetos.

Seguidamente, foram reunidas as *Frameworks* de CSS e mobile retiradas dos resultados das entrevistas com as *Frameworks* mencionadas no processo estudado na empresa. Foi possível concluir os seguintes dados, de acordo com a frequência com que as *Framework*s foram mencionadas:

- Bootstrap (4);
- Jquery Mobile (2);
- Foundation (2);
- Modernizr (1);
- Less (1);
- Sencha Touch (1);

Destes resultados anteriores nenhum dos entrevistados demonstrou incompatibilidades entre alguma das *Frameworks* mencionadas com a *Framework* javascript escolhida, contudo, o caso foi diferente durante o processo de decisão na empresa Prodcent, onde a junção da *Framework* de CSS Foundation com Angularjs não correspondeu ao esperado, também por questões de tempo e experiência do programador, e desta forma a *Framework* de CSS Foundation ter sido abandonada.

Por último, foi elaborada uma recolha de todas as *Frameworks* mencionadas ao longo das análises e é possível verificar esses resultados seguidamente na Tabela 6:

Framework	**Frequência**
AngularJs	3
Backbone.js	1
Jquery	4

Framework	Frequência
Knockout.js	2
Ember.js	2
CanJs	1
Prototype	1
YUI	1
JavascriptMVC	1

Tabela 6 - *Frameworks* mencionadas e frequência

Pela análise da tabela anterior pôde-se verificar que jQuery e Angularjs foram as *Frameworks* mais populares e que se destacaram perante os resultados analisados. Já Backbone, Canjs, Prototype, YUI e JavascriptMVC foram as menos relevantes nos resultados obtidos.

Anteriormente, no capítulo de enquadramento teórico, algumas das *Frameworks* que se podem observar na tabela anterior foram detalhadas segundo as suas principais características, contudo, outras, como Prototype, Yui e jquery não foram mencionadas neste documento, pelo que, nos próximos parágrafos, foi dada uma pequena descrição das mesmas para futuras referências.

- **Prototype**[61] **-** é uma Biblioteca orientada a objetos e foi desenhada para retirar complexidade da programação client-side, proporcionando APIs em volta de interfaces baseadas em Ajax e DOM. É suportada por *Ruby on Rails*, distribuída independentemente e usada por 2,9% de todos os sítios web, tornando-a uma das Bibliotecas de javascript mais populares. Objetiva o desenvolvimento de aplicações web dinâmicas permitindo fáceis e seguras alterações ao DOM. Tem como principais características o suporte avançado para manutenção de eventos, bastantes funcionalidades de Ajax e a extensão do DOM pela criação de novos elementos e métodos.
- **Jquery**[62] **-** é uma Biblioteca de javascript bastante popular e versátil. Tem como uma das principais características a sua compatibilidade com a grande maioria dos *browsers,* tornando-a a Biblioteca favorita entre 77% dos websites mais visitados mundialmente. A sua sintaxe foi desenhada para tornar mais simples a navegação pelo DOM, para criar animações, manipular eventos e desenvolver

[61] http://prototypejs.org/ - ultimo acesso: 23/09/2014

aplicações do estilo Ajax. É bastante flexível e rica em *plugins* facilitando a programação client-side.

- **YUI[63] -** é um Biblioteca de javascript e css destinada ao desenvolvimento de aplicações web interativas. Possui uma arquitetura modular tornando-a rápida e robusta. Desenvolvida por engenheiros da empresa Yahoo é uma Biblioteca intuitiva e com boa documentação comportando eventos de DOM e aplicações sustentáveis tanto em desktop, como dispositivos móveis e servidores. Utiliza técnicas como Ajax, DHTML e manipulação de DOM para o desenvolvimento de aplicações web interativas. Como referido anteriormente, a YUI é uma Bilbioteca de javasctipt e de CSS, uma vez que utiliza alguns recursos de CSS.

É de salientar que as 3 *Frameworks* descritas anteriormente não seguem o padrão de MVC e são consideras como Bibliotecas e não *Frameworks,* contudo, como justificado previamente no capítulo de *Frameworks* web, foram consideradas para avaliação dada a sua relevância na comunidade *online*, nomeadamente, de programação.

[62] http://jquery.com/ - ultimo acesso: 23/09/2014
[63] http://yuilibrary.com/ - ultimo acesso: 23/09/2014

VII. CAMINHOS DE DECISÃO

Após avaliadas as *Frameworks* era necessário compará-las com os critérios principais que se reuniram da análise dos dados das entrevistas, por forma a obter resultados relativos à *Framework* mais adequada segundo esses critérios.

Foram reunidos os principais critérios de escolha de uma *Framework*, de acordo com a frequência com que foram mencionados nas análises dos dados das entrevistas. Estes critérios foram dispostos na Tabela 7 e na Tabela 8 e foi dada uma pontuação a cada *Framework*, entre 0 (não satisfatório) e 3 (muito satisfatório), por cada um desses critérios.

A. Critérios de escolha principais

Na Tabela 3 no capítulo de análise dos dados das entrevistas foi possível visualizar o grupo de critérios referidos ao longo das análises. Esses critérios foram recolhidos de acordo com a frequência e importância com que foram mencionados e foram analisados por cada *Framework* até se obter uma *Framework* final. São os seguintes:

- **Comunidade** (3) – "*Existem muitos programadores a falar da Framework? É fácil encontrar soluções e fóruns a falar da Framework?*";
- **Documentação** (2) – "*Consigo entender facilmente todo o conteúdo referente à Framework no seu website? Esse conteúdo é legível para aprendizes ou demasiado técnico e específico?*";
- **Compatibilidade** (3) – "*A Framework é compatível entre os vários browsers e dispositivos existentes na atualidade? É compatível também com outras Frameworks?*";
- **Curva de aprendizagem** (3) – "*Quão complexa é a Framework e quanto tempo demorarei a aprender as suas funcionalidades?*";
- **Experiência** (2) – "*Já tenho conhecimento e prática com esta Framework.*";
- ***Plugins*** (2) – "*Existem muitas extensões e plugins que me possam auxiliar quando a Framework não disponibilizar a funcionalidade que necessitar?*";
- **Popularidade** (2) – "*A Framework já foi provada em mercado? É conhecida e utilizada por empresas de prestígio?*";
- ***Open source*** (1) – "*Tenho de pagar para poder usufruir da Framework?*";

- **Atualizada** (1) – "*A Framework encontra-se a par das tecnologias e funcionalidades da atualidade? Com que frequência evolui as suas versões para ser compatível à evolução da internet e dos browsers?*";
- **Funcionalidades** (1) – "*Qual o grau de ajuda que a Framework disponibiliza? Qual a quantidade de funcionalidades que tem e até que ponto deixa o programador usufruir dessas funcionalidades, poupando-lhe código e tempo?*";
- **Tamanho** (1) – "*Qual o tamanho da Framework por forma a não atrasar o carregamento inicial da minha página web?*";
- **Número de linhas de código** (1) - "*Da mesma aplicação, qual o número de linhas de código que foram necessários para a colocar a funcionar?* ";
- **Performance** (1) – "*A Framework suporta grandes quantidades de dados?*";
- **Maturidade** (1) – "*A Framework está estabilizada em mercado? É demasiado recente ou já foi desenvolvida há algum tempo?*";
- **Dependências** (1) – "*Para que a Framework inicialize são necessárias outras Frameworks? Para obter as funcionalidades da Framework na totalidade é preciso alguma dependência externa a essa Framework?*";
- **Flexibilidade** (1) – "*A Framework é facilmente adaptável com outras Frameworks? Integra-se com projetos já existentes e programados com outras Frameworks ou é demasiado específica, e só consegue seguir as suas próprias regras?*".

B. Escolha da *Framework* final

Seguidamente serão apresentados os resultados e as pontuações finais relativos aos critérios recolhidos com as *Frameworks* analisadas. Cada *Framework* foi estudada segundo um critério e dependendo da sua ligação forte ou não a esse critério em questão foi dada uma pontuação entre 0 e 3, como mencionado previamente. A pontuação dada a cada *Framework* foi refletida através de vários *websites,* livros e artigos de comparação das mesmas (Anderson, 2014; Buckler, 2014; Gruber, 2013; "How Complex are TodoMVC Implementations," 2013, "Javascript *Framework*s Comparison - Angular, Knockout, Ember and Backb…," 2014; Larson, 2012; McKeachie, 2013; Orsini, 2014; Porto, 2013; Shaked, s.d.; Shan, 2013; Trager & Kagan, 2013). Procuraram-se referências de conteúdo mais atual, por forma a ter resultados mais válidos e recentes, dada a evolução de cada *Framework* ao longo da sua existência.

	Angularjs	Backbone	Knockout	Ember	Canjs	Javascript MVC
Comunidade	3	3	2	2	1	2
Documentação	3	2	3	2	3	2
Compatibilidade	2	3	3	3	2	3
Curva de aprendizagem	2	3	3	1	3	3
Experiência[64]	3	1	0	0	0	0
Plugins	3	3	2	2	3	2
Popularidade	2	3	2	2	1	1
Open source	3	3	3	3	3	3
Atualizada	3	3	3	2	1	2
Funcionalidades	3	1	2	3	2	2
Tamanho	2	3	2	1	3	2
Número de linhas de código	3	2	3	3	3	2
Performance	2	3	3	2	3	3
Maturidade	2	3	2	2	2	3
Dependências	3	1	2	1	2	2
Flexibilidade	2	3	3	1	3	3
	41	**40**	**38**	**30**	**35**	**35**

Tabela 7 - Resultados das *Frameworks* vs. Critérios

[64] O critério da experiência foi baseado na experiência da investigadora para com as *Frameworks* selecionadas, os valores dados estão relacionados com a prática que a investigadora tem para com cada *Framework*.

	Jquery	Prototype	Yui
Comunidade	3	2	2
Documentação	3	3	3
Compatibilidade	3	2	3
Curva de aprendizagem	2	3	2
Experiência	2	0	0
Plugins	3	2	2
Popularidade	3	2	3
Open source	3	3	3
Atualizada	3	2	2
Funcionalidades	1	2	2
Tamanho	2	1	1
Número de linhas de código	2	2	2
Performance	2	2	2
Maturidade	3	2	3
Dependências	2	3	3
Flexibilidade	2	2	3
	39	**33**	**36**

Tabela 8 - Resultados das *Frameworks* vs. critérios

Pelos resultados visualizados nas tabelas anteriores pôde-se verificar que, a nível geral, todas as *Frameworks* estão muito próximas qualitativamente, destacando-se a *Framework* Angularjs pela maior pontuação e o EmberJs pela menor. Percorrendo critério a critério encontram-se algumas diferenças significativas que serão justificadas seguidamente de acordo com as referências recolhidas para os resultados visualizados.

Relativamente ao primeiro critério acerca da comunidade, as *Frameworks*/Bibliotecas Angularjs, Backbone e jQuery foram as que maior comunidade apresentaram e as que mais ativas se encontram atualmente a nível de *fóruns* de discussão, *reviews,* módulos, etc. Por estes motivos torna-se bastante acessível encontrar soluções para problemas que possam surgir quando o desenvolvimento de um projeto. Por outro lado, a Biblioteca CanJs destacou-se negativamente face a este critério, dificultando a sua possível escolha para um projeto.

A nível de documentação ambas as *Frameworks* apresentaram qualidade semelhante e com conteúdo bastante completo e simples, tanto nos seus *websites,* como em documentação externa. Destacaram-se como exceção o Backbone, Emberjs e JavascriptMvc que disponibilizam documentação um pouco mais confusa e escassa que as restantes. É de salientar que apesar de existir uma documentação ligeiramente

desorganizada para a Biblioteca Backbone, esta mostrou-se bastante completa no que respeita a bons exemplos e tutoriais para auxiliar os programadores. Também, respeitante ao critério de compatibilidade, todas as *Frameworks* pareceram se comprometer à disponibilização das suas funcionalidades pela grande maioria dos *browsers* e dispositivos. Apenas AngularJs, embora seja bastante compatível como as outras, demonstrou ser uma *Framework* muito moderna destinada a aplicações do tipo, logo tem mais dificuldades quando o funcionamento com *browsers* mais antigos, contudo, a própria *Framework* e a sua documentação disponibilizam técnicas e soluções para contornar esse inconveniente.

A curva de aprendizagem pode ser um dos critérios mais importantes para o desenvolvimento de um projeto, é esperado que a *Framework* escolhida corresponda às aprendizagens do programador e que este seja facilmente capaz de desenvolver código com a mesma. Knockout aparentou ser uma biblioteca bastante simples de aprender e dominar rapidamente, por outro lado, Ember mostrou ser a mais complexa dada a sua rigorosidade e por ser destinada a aplicações de maior complexidade. Angularjs pode ser compreendida pelas duas vertentes, demonstrou ser bastante acessível apreender os seus básicos para começar a desenvolver rapidamente, contudo, dada as suas novas terminologias para a linguagem HTML (diretivas, filtros, etc) implica que o programador tenha elevado domínio no DOM para compreender todos os elementos que Angularjs disponibiliza. Também toda a sua estrutura de modelos, *factories, providers,* serviços, pode ser confuso até o programador conseguir compreender cada conceito e qual o seu propósito.

Relativamente ao critério de *plugins* e extensões disponíveis para cada *Framework,* estas aparentaram ser bastante positivas na procura de funcionalidades extras a estas. Backbone destacou-se neste requisito, muito devido à sua simplicidade e popularidade que fazem surgir bastantes extras a esta, comportando funcionalidades que a biblioteca não disponha. Também jQuery é constituído por bastantes *plugins* para facilitar e compatibilizar as funcionalidades desta, contudo, muitas dessas extensões não são provadas em produção o que poderá tornar-se um inconveniente.

Como foi referido no parágrafo anterior, Backbone é uma biblioteca muito popular, senão a mais popular das restantes, tem um portfólio bastante completo e reconhecido, com *websites* tais como *Linkedin.com*, *twitter.com*, entre outros que se destacam pela sua abrangência nas redes sociais. jQuery também é muito notável dada a sua história e longevidade, uma vez que veio facilitar bastante a linguagem javascript ao apresentar conceitos novos, como animações, alterações diretas ao DOM, *date pickers,* etc. Yui, é

uma biblioteca que foi disponibilizada pela Yahoo, por esse aspeto já a torna bastante popular e com destaque perante as outras. JavascriptMvc e Canjs aparentaram ser as menos conhecidas.

Angularjs, embora ainda bastante recente, tem vindo a ganhar notoriedade rapidamente, já possui um portfólio bastante completo e demonstrou vir a ter cada vez mais sucesso ao longo do tempo. O facto de ser desenvolvida pela empresa Google, também lhe fornece estabilidade e firmeza. As imagens (Ilustração 9 e Ilustração 10) seguintes ilustram esta evolução gradual de interesse pela *Framework* Angularjs perante Backbone, jquery, Knockout e EmberJs (foram selecionadas estas quatro por terem demonstrado serem as mais populares segundo os dados recolhidos):

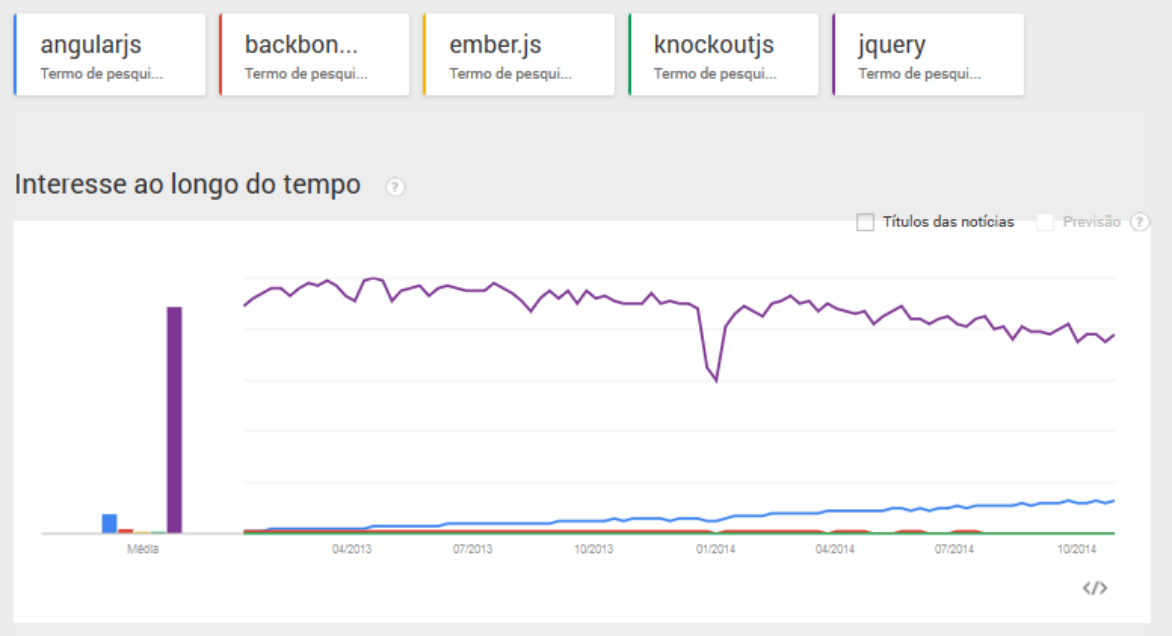


Ilustração 9 - Interesse global em resultados de pesquisa

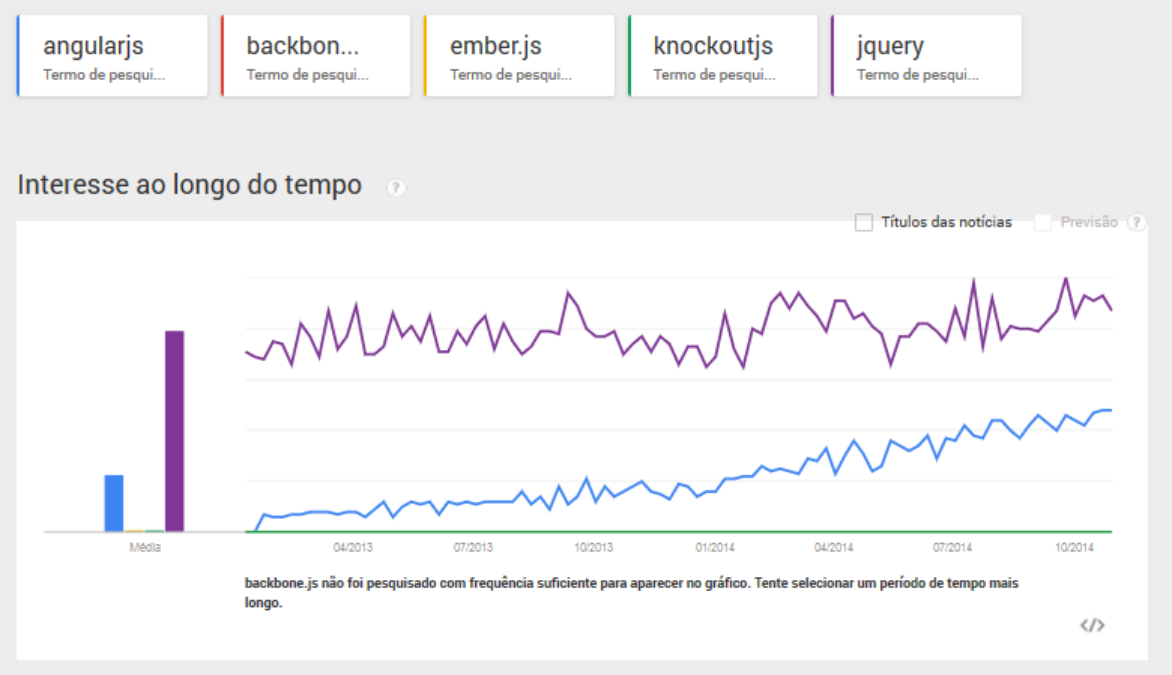


Ilustração 10 - Interesse global por vídeos no Youtube

O quão frequentemente é atualizada uma *Framework* é um aspeto importante para manter a mesma compatível com as evoluções dos *browsers* e das tecnologias virtuais. Canjs foi a ferramenta que demonstrou ser menos atualizada, sendo o seu conteúdo pouco avançado para o grau de desenvolvimento atual das restantes *Frameworks.* AngularJs, Backbone, jQuery e knockout aparentaram ser *Frameworks* bastante ativas e com atualizações frequentes das suas versões sempre com novas e inovadoras funcionalidades. Relativamente a essas funcionalidades, importava saber o que a *Framework* disponibiliza para se poder programar e reutilizar, em vez de se desenvolver novas funcionalidades de raíz. A *Framework* Angularjs e Ember aparentaram ser as *Frameworks* mais completas pelos dados recolhidos, contudo para Ember este aspeto pode tornar-se uma desvantagem uma vez que, lhe transmite pouca flexibilidade comparado com as outras. Para *Frameworks* demasiado simples como o Backbone, existe uma escassez de funcionalidades. Estas disponibilizam bastante liberdade ao programador de utilizar a *Framework* como desejar, contudo, o número de funcionalidades torna-se reduzido havendo necessidade de recorrer a mais dependências. Também jQuery carece de funcionalidades, é relativamente completo ao nível de modificações ao DOM, porém, torna-se pobre noutras funcionalidades também importantes como *routing, two way data binding,* entre outros.

O peso e o tamanho que uma *Framework* implica para um projeto é um aspeto que deve ser considerado para análise antes da seleção de uma *Framework*. É

importante ter em conta o tamanho da ferramenta no seu estado independente e o seu tamanho total em caso de necessidade de dependências extras. Prototype, Yui e Ember são consideradas as *Frameworks* mais pesadas para um *website* podendo causar atrasos no carregamento inicial das páginas web, porém, as duas primeiras atuam de forma independente, não necessitando de extras para poderem funcionar completamente, como é o caso da *Framework* Backbone, que apesar de ser bastante vantajosa no que respeita ao seu tamanho, é menos favorecida ao nível das dependências que necessita para o seu bom funcionamento (por exemplo: underscore.js e jquery/zepto).

O critério relativo ao número de linhas de código torna-se um pouco relativo devido à forma como um programador escreve o seu código e da lógica que utiliza para o mesmo, contudo, foi utilizada uma aplicação de teste do *website* TodoMvc e pôde-se concluir que a maioria das *Frameworks* encontram-se aproximadamente ao mesmo nível e número de linhas de código.

A Biblioteca Backbone, o jQuery, a Yui e o javascriptMvc são *Frameworks* já bastante estáveis e mais antigas o que as favorece ao nível da maturidade relativamente às restantes. São ferramentas com bastante experiência e histórico oferecendo a possibilidade de os programadores se sentirem mais confortáveis e seguros ao desenvolverem projetos com as mesmas.

A performance é uma consequência não só da *Framework* em si mas também da forma como é programada e são utilizadas as suas funcionalidades. Todas podem ter problemas de performance se a programação usada assim as sujeitar e o inverso também se verifica. Deste modo, as que poderão causar problemas mais rapidamente são as *Frameworks* Angularjs, Ember, jQuery, Prototype e Yui pelo facto de serem tão completas e disporem de inúmeras funcionalidades para ajudar o programador. Contrariamente a estas, Backbone e Knockout, pela sua simplicidade, são as que menos inconvenientes podem causar. Respetivamente à simplicidade destas *Frameworks,* este aspeto também as favorece ao nível do critério de flexibilidade, são facilmente integradas com outras *Frameworks* e projetos existentes. A estas duas *Frameworks* referidas também se juntam a Canjs, javascriptMvc e Yui. Emberjs, como referido, é a *Framework* mais complexa e mais completa o que a torna menos flexível, devido à sua capacidade de “obrigar” o programador a seguir os seus próprios padrões, sendo bastante “teimosa”.

Dos resultados do estudo anterior foi possível concluir que Angularjs é a *Framework* mais apta para o projeto, contudo, dada a pequena diferença entre as pontuações da Tabela 7 e Tabela 8 pôde-se concluir que tanto Backbone como Knockout

estão também favoravelmente inseridas nas melhores *Frameworks* mvc de javascript para o desenvolvimento do projeto em contexto empresarial.

É importante salientar que os requisitos do projeto são os critérios primordiais à seleção de uma *Framework*. Deste modo, foi elaborado um estudo relativamente aos requisitos do projeto perante as *Frameworks* e os critérios analisados. Era esperado nesta análise a exclusão de *Frameworks* à medida que forem deixando de ter interesse e vantagens para o projeto, e assim poder chegar a apenas algumas *Frameworks* finais, validando, por fim, se realmente a *Framework* Angularjs é a mais adequada ao projeto em desenvolvimento.

VIII. VALIDAÇÃO DA *FRAMEWORK* ESCOLHIDA

A. Requisitos do projeto

Como referido previamente, os requisitos do projeto são o aspeto primordial na seleção de uma *Framework*. É importante considerá-los antes de qualquer decisão ou avaliação, para não correr o risco de essa escolha ser dispendiosa para o projeto por decisões erradas ou precipitadas. Seguidamente serão apresentados os principais requisitos que o projeto centroproduto envolveu nas características de uma *Framework*. São eles:

- **Performance**: o projeto centroproduto é um *website* que necessita de suporte para grandes quantidades de dados e interações por parte dos seus utilizadores. É um projeto onde os dados podem ser constantemente atualizados, com necessidade de se manter consistência com todas as funcionalidades do *website.* É possível visualizar grandes listas de produtos, com imagens e informação relativa, assim como páginas para manutenção desses produtos com filtragens e ordenações. Estas funcionalidades precisam de ser desencadeadas em tempo real, sem necessidade de recorrer frequentemente a um servidor, e de modo a oferecer melhor usabilidade e experiência ao utilizador final;
- **Experiência do programador**: A experiência inicial do programador poderia ser um ponto favorável no desenvolvimento mais rápido do projeto se a *Framework* selecionada dispusesse de rápida aprendizagem e compreensão para o programador. Desta forma, seria muito mais rentável, a nível de custos de aprendizagem e mercado, desenvolver e programar o *website*;
- ***Routing***: devido à complexidade da plataforma *online*, um sistema de *routing* era importante para manter os seus conteúdos e endereços organizados e separados por conceitos. Este aspeto também disponibilizava uma melhor experiência ao utilizador, uma vez que os endereços criados seriam fáceis de ler, alterar e partilhar;
- **Comunidade, popularidade, documentação**: como se tratava de um projeto para desenvolver rapidamente, era importante recorrer a uma *Framework* conhecida, estável e com boa documentação para auxiliar o programador de forma rápida e eficaz, tornando possível, também, encontrar facilmente resultados persuasivos a problemas que pudessem surgir durante o desenvolvimento do projeto;

- **Curva e facilidade de aprendizagem**: Como já foi referido, pretendia-se desenvolver o projeto rapidamente, logo, a aprendizagem pela *Framework* teria de ser rápida e capaz o suficiente para se começar a desenvolver;
- **MVC**: como se tratava de um projeto algo complexo, era importante que o código se mantivesse sustentável e organizado para futuras reutilizações. Deste modo, era pretendido que a *Framework* escolhida seguisse em parte os padrões deste modelo;
- **SPA**: era pretendido desenvolver uma SPA, logo a *Framework* teria de disponibilizar funcionalidades para tal e ser facilmente adaptável a grandes atualizações parciais das páginas.

Após definidos os principais requisitos do projeto, foi elaborada uma nova tabela de pontuações, contudo, estas pontuações foram duplicadas, por forma a tornar as diferenças entre as *Frameworks* mais evidentes e por consequência, como se tratavam dos requisitos do projeto, estes tornaram-se mais importantes para a decisão final, tendo necessidade de serem destacados perante os critérios anteriores. Os critérios que não apresentaram qualquer vantagem na *Framework* foram pontuados negativamente diferenciando mais eficazmente as pontuações finais. Apenas foram selecionados os critérios relativos aos requisitos do projeto para a tabela seguinte; para aqueles que anteriormente já haviam sido pontuados na Tabela 7 e Tabela 8, foi duplicada a sua pontuação.

	Angularjs	Backbone	Knockout	Ember	Canjs	Javascript MVC
Comunidade	6	6	4	4	2	4
Documentação	6	4	6	4	6	4
Curva de aprendizagem	4	6	6	4	6	6
Experiência	6	2	-3	-3	-3	-3
Popularidade	4	6	4	4	2	2
Performance	4	6	6	4	6	6
Routing	6	6	-3	6	6	2
facilidade de aprendizagem	6	4	6	2	4	4
MVC	6	6	6	6	6	6
	48	46	32	31	35	31

Tabela 9 - *Framework*s vs. requisitos do projeto

	Jquery	Prototype	Yui
Comunidade	6	4	4
Documentação	6	6	6
Curva de aprendizagem	4	6	4
Experiência	4	-3	-3
Popularidade	6	4	6
Performance	4	4	4
Routing	2	4	6
facilidade de aprendizagem	6	6	4
MVC	-3	-3	-3
	35	28	28

Tabela 10 - *Frameworks* vs. requisitos do projeto

Pelos resultados da Tabela 9 e Tabela 10 confirmou-se que a *Framework* Angularjs é a mais adequada ao projeto, tanto pelos seus requisitos como pelos critérios avaliados no tópico anterior.

Descrevendo os requisitos do projeto por ordem de prioridades, destaca-se o critério de MVC que permite um código mais organizado e reutilizável dada a complexidade do projeto, que a longo prazo, irá tornar este critério quase obrigatório pelo aumento gradual das suas funcionalidades e potencialidades. Deste modo, analisando os resultados da tabela anterior, pôde-se verificar que as bibliotecas jquery, prototype e yui não se encontravam dentro dos padrões de *model-view-controller* logo não seriam adequadas ao projeto em desenvolvimento.

Seguiu-se o critério de performance do qual se destacaram as bibliotecas Backbone e Knockout pela sua simplicidade e tamanho, contudo, estas necessitam de outras bibliotecas para poderem funcionar por completo, logo este aspeto também poderá ser penoso para a performance de um *website*. Como foi mencionado previamente, a performance é um critério bastante relativo e depende essencialmente da forma como a programação é feita. Se a *Framework* for utilizada corretamente e sempre dentro dos seus padrões de boas práticas, a ocorrência deste tipo de problemas poderá ser consequentemente reduzida ou mesmo eliminada. Deste critério não foi possível excluir nenhuma *Framework,* sendo que, todas apresentaram características positivas relativamente a este critério, e, como referido, uma má performance não é de todo responsabilidade só da *Framework,* mas também da maneira como é utilizada.

A comunidade, a documentação e a popularidade foram os critérios que se seguiram com um grau de importância semelhante ao anterior. Destes critérios apenas se destacou a *Framework* Canjs pela menor comunidade e popularidade, no entanto, pelos resultados obtidos, aparentou ser uma das melhores *Frameworks* a nível de

documentação. Ambas as *Frameworks* restantes se apresentaram com boa qualidade perante estes critérios, e deste modo, não foi possível excluir nenhuma, uma vez que todas demonstraram eficácia e eficiência nestes critérios. Contrariamente aos requisitos anteriores, no critério da experiência do programador já foram destacadas algumas *Frameworks* que não preenchem estes requisitos, são elas o Knockout, EmberJs, Canjs e javascriptMvc, sobrando, deste modo, Backbone e Angularjs.

O critério de *routing* foi o que se seguiu na ordem das prioridades para o projeto e pôde-se concluir que, para este requisito, ambas as *Frameworks* Angularjs e Backbone foram favorecidas, tendo as duas a pontuação mais elevada (6). Finalmente, o requisito, também prioritário, mas com menos relevância que os anteriores, foi a curva e a facilidade de aprendizagem da qual resultou um empate entre as duas *Frameworks* restantes. Embora a *Framework* Angularjs aparentasse ser mais fácil de aprender, a Biblioteca Backbone provou ter uma curva de aprendizagem maior.

A Framework AngularJs é uma *Framework* que demonstrou encontrar-se em ascensão aparentando estar a evoluir cada vez mais na comunidade de programadores. Também o facto de ser desenvolvida por elementos da empresa Google transmite-lhe mais credibilidade e estabilidade. O seu número ilimitado de funcionalidades, contrariamente à Biblioteca Backbone, favorece-a positivamente, necessitando apenas de melhorar a sua documentação perante funcionalidades mais complexas. É uma *Framework* bastante isolada e independente, não necessitando de extras para funcionar completamente como é o caso da Bilblioteca Backbone.

Embora a curva de aprendizagem da *Framework* AngularJs possa sofrer altos e baixo, é uma curva sempre gradual até se dominar por completo. Como comprovado anteriormente também, é uma *Framework* ideal para SPA e CRUD, o que para o projeto em destaque é bastante importante, uma vez que a constante atualização (criação, remoção, edição) de dados e formulários é uma das funcionalidades mais imprescindíveis do *website.*

Tanto a *Framework* Angularjs como a Biblioteca Backbone seriam adequadas ao projeto, contudo, pelos motivos supracitados e pela urgência do projeto inicialmente, uma vez que a *Framework* AngularJs é mais acessível e rápida de aprender, conduziram a que esta *Framework* fosse a ideal no projeto centroproduto. Seguidamente será apresentada a validação da *Framework* Angularjs perante alguns elementos da empresa Prodcent acerca das suas experiências com esta no projeto a decorrer.

B. Recolha e análise de dados

1. Análise da entrevista

Por forma a validar a utilização da *Framework* AngularJs na empresa Prodcent, foi elaborada uma entrevista a alguns elementos da mesma para se poder obter algum feedback, tanto positivo como negativo, da experiência destes elementos com a *Framework.*

O objetivo primordial desta entrevista foi conhecer a opinião e a experiência dos entrevistados perante a *Framework*. Não era pretendido obter respostas demasiado técnicas, apenas uma descrição daquilo que a *Framework* transmitiu aos entrevistados.

Tratou-se de uma entrevista exploratória do tipo livre ou semiestruturada, uma vez que, tal como na primeira entrevista, procurava-se obter relatos das experiências dos entrevistados, logo não poderia ser algo demasiado objetivo, que os impedisse de se expressarem abertamente.

Foram elaboradas 7 entrevistas seguidas por um guião, objetivado a manter o rumo e a organização aquando o momento da mesma (Anexo 4 – Guião de entrevista aos elementos da empresa). As fontes de informação variaram entre programadores, gestores e *testers (*utilizadores finais). A diversidade das fontes foi propositada por forma a se ter uma abordagem global por vários géneros de público-alvo: os programadores por realizarem o seu código com a *Framework*, o gestor por ter permitido a sua utilização e desta forma poder calcular custos de produção e consequências dessa decisão, e o *tester* que, neste caso, ao testar a plataforma, é como se se tornasse num utilizador final dando um feedback mais funcional e *User Experience* (UX) do uso do *website* programado com a *Framework* em questão. O registo dos dados das fontes de informação foi todo elaborado através de texto.

Seguidamente, na Tabela 11, será visualizada a caracterização das fontes de informação segundo os critérios de área de formação e papel que desempenham na empresa Prodcent. A cada entrevistado foi dado uma referência empírica para futuras citações ao longo do documento:

Entrevistado	Área de formação	Papel que desempenha na empresa
E1	Engenharia informática	Programador back-end
E2	Novas Tecnologias da Comunicação	Programador front-end
E3	Novas Tecnologias da Comunicação	Web *designer*
E4	Matemática	Programador fron-end
E5	Engenharia Computadores e Telemática	*Tester (*utilizador final)

Entrevistado	Área de formação	Papel que desempenha na empresa
E6	Matemática	Programador back-end
E7	Engenharia Eletrotécnica	Gestor de projeto

Tabela 11 - caracterização das fontes de informação

Na análise dos dados da entrevista procurou-se identificar os principais aspetos positivos e negativos das reflexões dos entrevistados. A entrevista encontrava-se estruturada em duas vertentes, sendo a primeira relativa ao conhecimento do entrevistado com a *Framework* Angularjs e a segunda referente à validade da mesma para o projeto.

Para a primeira parte da entrevista procurou-se conhecer se os entrevistados já haviam tido experiências anteriores com a *Framework* e quais foram essas experiências por forma a conhecer o nível de maturidade que o entrevistado admitia em relação à *Framework,* tornando desta maneira, os dados mais completos com entrevistados mais experientes. Seguidamente, foram selecionados alguns critérios, escolhidos de acordo com os requisitos principais do projeto e que fossem de fácil entendimento para os entrevistados. São eles, a performance, comunidade, curva de aprendizagem, documentação e popularidade. Para esses critérios foi requerido aos entrevistados que caracterizassem a *Framework* de acordo com a sua opinião e experiência acerca dos mesmos. A cada critério foi dada uma pequena explicação para o entrevistado poder saber o que deveria mencionar em cada um. Por último, neste ponto, foi também pedido aos entrevistados que detalhassem os aspetos mais positivos e negativos que consideravam acerca da *Framework.*

Na segunda parte da entrevista, foi importante conhecer a opinião dos entrevistados sobre a *Framework* dentro do projeto, a sua validade e utilidade. Inicialmente procurou-se saber quais os principais convenientes e inconvenientes que a *Framework* tem demonstrado no projeto em desenvolvimento e qual a sua utilidade para o mesmo, se o entrevistado a considera vantajosa e porque motivos. Procurou-se também conhecer, dos entrevistados, a experiência que têm com a *Framework* no projeto a decorrer. Neste ponto nem todos os entrevistados terão tido uma experiência direta com a *Framework,* pelo que as respostas foram essencialmente dadas por parte dos programadores do projeto. Por fim, procurou-se identificar se ambos os entrevistados teriam mais algum aspeto a acrescentar que quisessem partilhar.

Seguidamente, nos próximos parágrafos, serão descritos os dados recolhidos pelas entrevistas, por forma a validar a *Framework* e confirmar a sua utilidade para o projeto em desenvolvimento.

2. Análise dos dados

Durante o processo de análise dos dados das entrevistas procurou-se identificar, inicialmente, se os entrevistados já haviam experimentado a *Framework* antes de o projeto iniciar, a resposta foi semelhante entre todos e refletiu-se negativamente, nenhum dos entrevistados havia utilizado, ou tinha qualquer conhecimento da *Framework* antes do início do desenvolvimento do *website*. Seguidamente foram enunciados 5 critérios pelos quais se esperou que os entrevistados caracterizassem a *Framework* conforme cada critério. Os resultados foram os seguintes:

- Critério de **comunidade:** A maioria dos entrevistados concordou que é uma *Framework* com bastante comunidade e muito recente, que se tem destacado ao longo do tempo tornando-se cada vez mais utilizada facilitando a procura de soluções a problemas, novos módulos, entre outros. Foi ainda mencionado por um dos entrevistados que se nota uma grande adesão de colaboradores da *Framework,* o que se reflete também na quantidade de pessoas que participam em discussões sobre a mesma. Em suma, pôde-se concluir que a *Framework* Angularjs, embora recente ainda, tem evoluído bastante trazendo cada vez mais comunidade e ajuda a quem interage com a mesma.
- Critério de **curva de aprendizagem:** Relativamente a este critério, grande parte das respostas foram concordantes no que respeita ao facto de a *Framework* AngularJs ser muito simples de aprender e começar a trabalhar com ela, é bastante acessível obter noções básicas da *Framework* e esse aspeto é suficiente para se começar a desenvolver um projeto. Contudo, três dos entrevistados concordaram, que embora fácil de aprender inicialmente, a *Framework* pode-se tornar complexa quando se pretende aproveitar todo o conjunto de funcionalidades que esta disponibiliza, sendo necessário algum esforço por parte do aprendiz. Deste modo, pôde-se confirmar que a *Framework* é bastante rápida de aprender e dominar o seu modo geral de funcionamento, no entanto, torna-se mais complexa à medida que se vai aprofundando as suas funcionalidades.
- Critério de **documentação:** A maioria dos entrevistados concordou que a documentação disponibilizada pela *Framework* era bastante simples e organizada com bons tutoriais de exemplo, auxiliando qualquer aprendiz a aprender a *Framework* autonomamente. Também foi concordante que a documentação se encontrava bastante detalhada e com muitos exemplos para facilitar a aprendizagem. Apenas um dos entrevistados mencionou que nem sempre a documentação é suficiente. Desta forma, foi possível concluir que a

documentação disponibilizada pela *Framework* é suficiente para uma boa aprendizagem inicial, sendo possível visualizar exemplos e realizar alguns tutoriais caso necessário.

- Critério de **performance:** Este foi o critério mais criticado pelos entrevistados, considerado até o mais desvantajoso. A maioria dos entrevistados referiu que a performance pode realmente ser um problema para um *website* se as suas funcionalidades não forem usadas corretamente. E1 relatou que esses problemas de performance podem resultar de uma má utilização dessas funcionalidades, e que, uma boa utilização das mesmas pode resultar no oposto que se afirmou no início deste ponto. E2 acrescentou ainda que se deve ter em atenção ao que se quer atualizar constantemente, conhecer o que poderá variar quando se navega e dividir esse conteúdo entre o que a *Framework* Angularjs deve "observar" e o que não é necessário. Referiu ainda que, como qualquer outra *Framework*, é necessário ter em atenção as facilidades que estas disponibilizam e não "abusar" em demasia das mesmas. E5, como sendo um utilizador final, destacou para o facto de o *browser,* por vezes e após algum tempo, ficar mais lento quando se situa no *website* desenvolvido com Angularjs, consumindo bastante memória e processamento. Por último E6 terminou dizendo que por ser uma *Framework* 100% *client-side* pode dar problemas de performance em projetos de grandes dimensões, dependendo sempre da organização e divisão das funcionalidades. Acrescentou ainda que "*Umas das principais vantagens da Framework que é o two way data binding, apresenta problemas de performance para volumes de dados superiores a duas mil unidades, como previsto pelos autores da Framework"*. Pôde-se concluir, assim, que o critério de performance pode ser problemático para esta *Framework* se não for efetuada uma boa utilização das funcionalidades da mesma.
- Critério de **popularidade:** A maioria dos entrevistados, concordou, para este critério, que Angularjs é uma *Framework* bastante recente, não sendo, por este motivo, utilizada por muitas empresas de grande prestígio, contudo, também concordaram que, pelo facto de ser desenvolvida pela empresa Google lhe dá bastante credibilidade e reconhecimento. E2 acrescenta também que "*ainda é uma Framework recente mas aparenta ter grandes potencialidades e não tardará a ter cada vez mais 'adoradores"*. Pôde-se concluir, desta forma, que a popularidade da *Framework* Angularjs se encontra a evoluir exponencialmente e

que dada a sua qualidade e colaboradores (Google) lhe poderá dar grande prestigio futuramente.

Antes de se iniciar a segunda parte da entrevista, referente à validação da *Framework* no projeto centroproduto, foi pedido aos entrevistados que enunciassem os aspetos que consideravam mais positivos e negativos acerca da *Framework.* Como consequência, a Tabela 12 representa os principais aspetos mencionados pelas fontes de informação:

	Nº de ocorrências	**Conceito**
Pontos positivos:	3	Aprendizagem muito rápida;
	1	Boa performance;
	2	Desenvolvimento rápido;
	2	*Two way data binding;*
	1	Criação de diretivas personalizadas;
	1	*Dependency injection;*
	2	Boa documentação;
	2	Boas funcionalidades;
	1	Compatibilidade com dispositivos móveis;
	1	Dinamismo ao HTML;
	1	Boa UX (*User Experience);*
	1	Acesso web similar a programas nativos;
	1	Compatibilidade com a empresa Google;
	1	Evolução rápida;
Pontos negativos:	1	Recente;
	3	Performance;
	1	Curva de aprendizagem por ter muitos conceitos novos de HTML;
	1	Pouco flexível com outras *Frameworks*;
	1	Falta clareza em pormenores avançados da *Framework* que por vezes tornam o *debugging* complicado;
	1	Programação menos bem planeada/estruturada;
	1	Programação com utilização de memória algo descontrolado.

Tabela 12 - Pontos positivos e negativos da *Framework* Angularjs

Pelos resultados anteriores, foi possível concluir que a performance foi considerada a maior desvantagem da *Framework* Angularjs, tendo o maior número de ocorrências. Por outro lado a facilidade de aprendizagem, o desenvolvimento rápido, o

two way data binding, e as funcionalidades, foram os aspetos considerados mais vantajosos para a *Framework* e que tiveram maior relevância entre os entrevistados.

A segunda parte da entrevista foi mais direcionada à utilidade da *Framework* para o projeto em desenvolvimento na empresa. Inicialmente, procurou-se compreender as principais vantagens e desvantagens que a *Framework* transmitia ao projeto. As respostas dadas pelos entrevistados foram bastante distintas e cada um relatou a sua opinião conforme a experiência que obteve com a mesma. E1 mencionou que as funcionalidades da *Framework* são muito úteis ao projeto, para aspetos como pesquisas e filtros que são muito utilizados no *website*, a *Framework* disponibiliza funcionalidades que permitem tornar essas funções muito mais compreensíveis. Também indicou que em formulários, a *Framework* Angularjs é muito útil com as suas validações, uma vez que permite que o utilizador final não se engane tão facilmente, dando-lhe rápido feedback e soluções. Como desvantagens referiu que por ser processada do lado do cliente, quando muitos processos estão a ser executados ao mesmo tempo, pode tornar o *browser* mais lento e com *leaks* de memória. O entrevistado E2 referiu a importância que uma *Framework SPA* abriga no projeto, dado o seu dinamismo e grande quantidade de partes da página que devem ser atualizadas constantemente, facilitando também a experiência do utilizador. Ao nível da programação acrescentou que esta *Framework* simplifica bastante o trabalho, podendo manter-se o código organizado e separado logicamente pelas suas funcionalidades. Mencionou apenas uma desvantagem referente à funcionalidade *ng-repeat* que poderá trazer inconvenientes de performance para grandes listas, como é o caso do projeto, que contem muitas listas de produtos, sendo esse o foco principal do mesmo. Contudo, acrescentou que esse pormenor está a ser improvisado devido à nova funcionalidade de *track by* que permite comunicar ao Angularjs o que deve ser atualizado ou não, conforme alterações no *track by,* e desta forma, a *Framework* não necessita de estar constantemente a "*observar*" os elementos dessas listas.

Ainda nas vantagens e desvantagens da *Framework* para o projeto, E3 indicou que a performance, a economia de custos e a compatibilidade entre os *browsers* são aspetos bastantes positivos da *Framework* e que colocam o projeto numa boa posição, também por a mesma ser bastante conhecida e utilizada. E4, tal como E2, também referiram que a utilização da *Framework* facilita a programação do código *front-end* da plataforma. E5, como utilizador final, mencionou a vantagem a nível de interação do utilizador, contudo, reforçou que pode tornar o *website* lento após muito tempo de utilização. E6 salientou a facilidade com que se consegue separar o *front-end* e o *back-end* utilizando a *Framework* Angularjs para englobar toda a aplicação *front-end*. Referiu

ainda, que a dificuldade de trabalhar o SEO é uma das grandes desvantagens da mesma. Por último E7, como gestor do projeto, mencionou que a *Framework* proporciona uma modernidade distintiva ao projeto. Em suma pôde-se concluir que, embora as opiniões descritas anteriormente sejam bastante diversificadas, destacou-se o facto de a nível de UX, funcionalidades e simplicidade na programação, a *Framework* disponibiliza bastantes vantagens ao projeto. Por outro lado, como já foi referido, a performance manteve-se como sendo a desvantagem principal da *Framework* para o projeto.

Seguidamente, foi requerido aos entrevistados que descrevessem a utilidade da *Framework* no projeto em desenvolvimento. De um modo geral, todos os entrevistados concordaram que a *Framework* é bastante útil ao trabalho. Alguns entrevistados fizeram referência às funcionalidades desta e como estas se enquadram nos requisitos do projeto, outros indicaram que o facto de esta *Framework* tornar o HTML tão dinâmico e estrutural ao mesmo tempo, facilita bastante o trabalho dos três programadores de *front-end*. Também, a experiência do utilizador foi mencionada com ponto positivo e útil da *Framework* Angularjs, assim como o suporte para grandes quantidades de dados, sendo este um dos requisitos principais do *website* centroproduto. Por último, foi referido também que a *Framework* permite desenvolvimentos *front-end* complexos e rápidos, com alguns intervalos destinados à otimização e testes da programação.

Por fim, o ultimo aspeto que interessou conhecer dos entrevistados foi a sua experiência com a *Framework* ao longo do projeto, tanto a nível de programação como utilização. Como se trata de uma equipa bastante diversificada, as experiências que se obtiveram foram bastante distintas. E1 referiu que, apesar de ter aprendido só os básicos da *Framework*, esta é bastante simples de trabalhar, E3 também concordou com o facto de estar em fase de aprendizagem ainda, mas que até agora está a correr de forma positiva. E2 mencionou que ainda tem muito para aprender mas que pela experiência que obteve até agora pode garantir que a *Framework* lhe facilitou bastante o trabalho, sendo muto intuitiva e rápida de aprender. E4, tendo recentemente começado a trabalhar com a *Framework,* refletiu o facto de não estar a ser complicado utilizar a mesma. E5, como sendo o *tester,* e, neste caso, o representante de um utilizador final mencionou que a *Framework* é muito útil tornando mais fácil e rápida a interação do utilizador. E6 teve contacto com a *Framework* apenas no início do desenvolvimento do projeto, utilizando as funcionalidades mais básicas da mesma, contudo, mencionou que na altura lhe pareceu ser bastante complicado de aprender alguns aspetos mais avançados, como por exemplos as *diretivas.* Também acrescentou que a compreensão do modelo MVC, para esta *Framework*, não foi fácil principalmente em termos de estruturação tanto do código

como dos ficheiros. Por fim, E7 como gestor do projeto e sendo este que haverá tomado a decisão final acerca da escolha da *Framework* Angularjs considerou que foi uma boa decisão e que ainda existe muito caminho a percorrer para aprender a trabalhar mesmo bem com a *Framework.* Pôde-se concluir que a experiência dos entrevistados ainda é um pouco escassa, havendo muitos casos em que apenas existe um conhecimento mais básico, contudo, pelos resultados obtidos, foi possível concluir, também, que as experiências são positivas e a vontade de aprender mais destacou-se, tal como acrescentou, no final da sua entrevista, a fonte de informação E5.

Em suma foi possível confirmar que a *Framework* Angularjs é a mais adequada ao projeto tanto pelos requisitos do projeto como pelas opiniões dos seus programadores, utilizadores e gestores. Ambos os resultados revelaram aspetos vantajosos à *Framework* e que a escolha final não foi em vão, poupando custos extras e tempo.

IX. CONCLUSÕES

Após todo o processo de escolha é possível afirmar que o elevado número de *Frameworks* existentes atualmente interfere na decisão final de qualquer programador ou gestor, tornando todo este processo, muito mais complicado e exaustivo. A importância da necessidade de seguir uma metodologia no processo de seleção de uma *Framework* torna-se cada vez mais evidente e não abrange apenas os programadores de uma empresa mas inclui os próprios gestores.

De um modo geral, pode-se assumir que os objetivos, tanto específicos como principais, foram cumpridos com sucesso, não havendo dificuldades nas várias etapas que abrangiam todo o processo de desenvolvimento. Também a recolha de dados da primeira entrevista se destacou positivamente, os resultados da análise desses dados foram suficientes para uma boa análise que serviu de guião para o restante processo de escolha da *Framework.*

Os critérios que foram considerados mais importantes no final da investigação foram a comunidade, a documentação, a compatibilidade, a curva de aprendizagem, a experiência do programador, os *plugins*, a popularidade, *open source*, atualização, as funcionalidades, o tamanho, o número de linhas de código, a performance, a maturidade, as dependências, e por último a flexibilidade. Todos estes critérios devidamente analisados por todas as *Frameworks* selecionadas para a escolha final, puderam resultar numa *Framework* final denominada Angularjs que se destacou pelas suas qualidades e por corresponder aos requisitos essenciais para o projeto em contexto empresarial.

Os resultados perante a *Framework* Angularjs foram bastante compensadores, tanto a nível do processo de investigação como do projeto em contexto empresarial, uma vez que a *Framework* correspondeu à escolha já antes elaborada na empresa Prodcent. A Angularjs tem vindo a ser destacada nas comunidades *online*, principalmente nas comunidades de programadores, pelas suas qualidades e funcionalidades, indiciando que poderá vir a ser cada vez mais utilizada.

Embora os dados recolhidos ao longo deste trabalho demonstrem que a *Framework* Angularjs foi a mais adequada ao projeto desenvolvido, estas conclusões não podem ser generalizadas. No processo de seleção de uma *Framework,* para além das características desta, devem ser tidas em consideração as características do projeto. Ou seja, a *Framework* com melhor pontuação nos critérios mais importantes não significa que seja a *Framework* ideal para todos os projetos. *Backbone*, Knockout e jQuery

também se destacaram pelas suas qualidades e poderão dar bastantes vantagens a outras tipologias de projeto.

Ao longo desta investigação procurou-se encontrar uma resposta à questão de investigação formulada inicialmente. Esta questão procurava conhecer os principais critérios de seleção de uma *Framework* mvc para o desenvolvimento de aplicações *front-end* c*lient-side* web ou *mobile*, e pelos resultados obtidos, pode-se concluir, finalmente, que os requisitos do projeto, assim como as funcionalidades que lhe estão adjacentes são os principais critérios quando o início do processo de seleção de uma *Framework,* são também os que têm maior prioridade perante todos os restantes que se listaram durante toda o processo de desenvolvimento.

A. Limitações

Umas das maiores limitações no desenvolvimento de trabalhos de investigação nesta área está associada à rápida e continua evolução das tecnologias. Esta evolução não é por si só uma falha na investigação, mas sim uma consequência da realidade da qual poderá condicionar as conclusões finais.

A realização da entrevistas que não se efetuaram de forma presencial também podem ser consideradas uma limitação, uma vez que pode ter impedido os entrevistados de se expressarem mais livremente ou descreverem melhor as suas experiências. Também relativamente às entrevistas, no caso das segundas, pode-se considerar que a escassa experiência e o pouco conhecimento dos entrevistados perante a *Framework* AngularJs refletiu possivelmente, para resultados mais básicos e incompletos, contudo, embora possa ser considerada uma limitação, não influenciou os resultados finais.

O processo que se considerou ideal para o desenvolvimento deste trabalho de investigação não foi possível seguir na totalidade devido a este ter sido desenvolvido em âmbito empresarial, o que obrigou a uma adaptabilidade às metodologias e regras da empresa e do processo de desenvolvimento do projeto.

B. Trabalho Futuro

Um dos trabalhos a realizar futuramente poderá estar associado a uma pequena aplicação que efetue os caminhos de decisão desde os critérios de escolha até à *Framework* mais adequada, por forma a auxiliar programadores e gestores nas decisões para os seus projetos. Validar os critérios de seleção em outras tipologias de projetos

também é um fator que pode ser abordado futuramente, oferecendo mais generalidade a este documento e não se focar apenas num tipo de projeto.

Como o conceito de *Framework* está em constante atualização, havendo cada vez mais e melhores *Frameworks* a necessidade de analisar mais *Frameworks* será um trabalho constante que poderá ser revisto futuramente, assim como as características que foram mencionadas para cada *Framework* analisada necessitarem de ser atualizadas à medida que as tecnologias evoluem. Também uma análise mais técnica poderá ser um trabalho a elaborar futuramente, a elaboração de protótipos funcionais com cada *Framework* para obter resultados mais práticos e não apenas baseados em experiência e opinião de programadores e gestores de projeto.

X. BIBLIOGRAFIA

10 Reasons Web Developers Should Learn AngularJS. (2013). Acedido em Agosto 14, 2014, em http://www.wintellect.com/blogs/jlikness/10-reasons-web-developers-should-learn-angularjs

A linguagem XML. (s.d.).

AJAX Introdução. (s.d.). Acedido em Agosto 03, 2014, em http://www.w3schools.com/ajax/ajax_intro.asp

Allen, S., Graupera, V., & Lundrigan, L. (2010). *Pro Smartphone Cross-Platform Development*. Berkeley

Anderson, M. (2014). AngularJS vs Ember.js. Acedido em Setembro 24, 2014, em http://thoughts.z-dev.org/2014/06/26/angular-js-vs-ember-js/

Anil, N. (2013). Exploring JavaScript MV* *Framework*s Part 1 – Hello Backbonejs. Acedido em Julho 03, 2013, em http://www.infragistics.com/community/blogs/nanil/archive/2013/04/01/exploring-javascript-mv-*Framework*s-part-1-hello-backbonejs.aspx

Backbonejs vs Angularjs : Demystifying the myths. (2013). Acedido em Julho 03, 2013, em http://www.nebithi.com/2012/12/27/backbone-and-angular-demystifying-the-myths/

Barnard, E. M. (2012). *KnockoutJS Starter* (p. 1–51).

Bednarski, W. (2013). *Learning JavaScriptMVC* (p. 1–124).

Bodmer, M. (2013). *Instant Ember . js Application Development How-to* (p. 1–48).

Buckler, C. (2014). What the Death of YUI Can Teach Developers. Acedido em Setembro 25, 2014, em http://www.sitepoint.com/death-yui-can-teach-developers/

Clarke, A. (2013). *Sass and Compass for Designers* (p. 1–274).

Clifton, M. (2003). What Is A *Framework*? Acedido em Novembro 15, 2013, em http://www.codeproject.com/Articles/5381/What-Is-A-*Framework*

Coutinho, C. P. (2011). *Metodologias de Investigaçao em Ciências Sociais e Humanas* (p. 337).

Coutinho, C. P., Sousa, A., Dias, A., Bessa, F., Ferreira, M. J., & Vieira, S. (2009). Investigação Acção Metodologias. *Psicologia, Educaçao E Cultura*, *13*(2), 455–479.

Cummings, C. (2012). Diving into CanJS | Nettuts+. Acedido em http://net.tutsplus.com/tutorials/javascript-ajax/diving-into-canjs/

Dwyer, S. (2009). Progressive Enhancement: What It Is, And How To Use It? | Smashing Coding. Acedido em Dezembro 21, 2013, em http://coding.smashingmagazine.com/2009/04/22/progressive-enhancement-what-it-is-and-how-to-use-it/

Engström, A., & Salehi-Sangari, E. (2007). *Assessment of Business-to-Business (B2B) e-Marketplaces' Performance*. Luleå University of Technology. Acedido em http://epubl.ltu.se/1402-1544/2007/22/LTU-DT-0722-SE.pdf

Firtman, M. (2012). *jQuery Mobile: Up and Running*. (S. st. Laurent, Ed.) (1st ed., p. 245). USA: O´Reilly Media.

Flanagan, D. (s.d.). JavaScript: The Definitive Guide, 4th Edition. O'Reilly & Associates, Inc. Acedido em http://docstore.mik.ua/orelly/webprog/jscript/ch11_03.htm

Georgiev, M., Jana, S., & Shmatikov, V. (2014). Breaking and Fixing Origin-Based Access Control in Hybrid Web / Mobile Application *Framework*s, *14*, 23–26.

Graziotin, D., & Abrahamsson, P. (2013). Making Sense Out of a Jungle of JavaScript *Framework*s: towards a Practitioner-Friendly Comparative Analysis. In *14th International Conference on ProductFocused Software Process Improvement PROFES 2013* (p. 334–337)

Graziotin, D., Abrahamsson, P., & Osmani, A. (2013). Making Sense Out of a Jungle of JavaScript *Framework*s: towards a Practitioner-Friendly Comparative Analysis. In *Essentials, Javascript, MVC* (p. 334–337)

Gruber, D. (2013). The 10 hottest JavaScript *Framework* projects. Acedido em Setembro 24, 2014, em http://www.infoworld.com/article/2612250/application-development/the-10-hottest-javascript-*Framework*-projects.HTML

Grüneberger, F. J. (2012). *REAL-TIME COLLABORATION SUPPORT FOR JAVASCRIPT*.

Hales, W. (2012). HTML*5 and Javascript Web Apps* (p. 1–171).

How Complex are TodoMVC Implementations. (2013). Acedido em http://blog.coderstats.net/todomvc-complexity/

Javascript *Framework*s Comparison - Angular, Knockout, Ember and Backb…. (2014). Acedido em September 23, 2014, em http://pt.slideshare.net/deepusnath/javascript-*Framework*s-comparison-angular-knockout-ember-and-backbone

Knight, K. (2011). Responsive Web Design : What It Is and How To Use It, 1–40.

Kozlowski, P., & Darwin, P. B. (2013). *Mastering Web Application Development with AngularJS*. (R. Khambatta & P. Balan, Eds.) (1st ed., p. 1–372). UK: Packt Publishing Ltd.

landland. (2013, April 11). Backbone vs Ember vs Angular vs Knockout Review. Acedido em Julho 17, 2013, em http://www.martywong.com/blog/backbone-vs-ember-vs-angular-vs-knockout-review

Larson, R. (2012). jQuery: the good, the bad & the ugly. Acedido em Setembro 24, 2014, em http://www.webdesignerdepot.com/2012/09/jquery-the-good-the-bad-and-the-ugly/

Lau, R. Y. K. (2007). Towards a web services and intelligent agents-based negotiation system for B2B eCommerce. *Electronic Commerce Research and Applications*, *6*, 260–273. Acedido em http://ac.els-cdn.com/S1567422306000433/1-s2.0-

S1567422306000433-main.pdf?_tid=90e232fa-47e2-11e3-84cc-00000aab0f6c&acdnat=1383852597_113e4e3be4c9c5c9e8f852285fd4de28

Marcotte, E. (2011). 11 Reasons why responsive web design isn't that cool! Acedido em Julho 17, 2013, em http://www.webdesignshock.com/responsive-design-problems/

Marriner, E. (s.d.). MVC JavaScript Applications, 1–37.

McFarland, D. S. (2012). *CSS3 the missing manual*.

McKeachie, C. (2013). Choosing a JavaScript MVC *Framework*. Acedido em September 23, 2014, em http://www.funnyant.com/choosing-javascript-mvc-*Framework*/

Mikkonen, T., & Taivalsaari, A. (2007). Web Applications - Spaghetti Code for the 21st Century. *Sun Microsystems*, 1–25.

Miller, H. G. (2011). The spark of innovation begins with collaboration. *Noblis*, *11*(1), 1–56.

Moraes, J., Breda, M., Gil, P., & Medaglia, R. (s.d.). Web Services.

Musser, J. (s.d.). *Web 2.0 Principles and Best Practices* (p. 0–9). O'Reilly Media, Inc.

Nicollet, V. (2012). HTML rendering : client or server ? Acedido em Julho 10, 2013, em http://nicollet.net/2012/07/HTML-rendering-client-or-server

O'Dell, J. (2011). Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app. Acedido em Dezembro 20, 2013, em http://venturebeat.com/2011/08/16/linkedin-node/

Orsini, L. (2014). Angular, Ember, And Backbone: Which JavaScript *Framework* Is Right For You? Acedido em Agosto 14, 2014, em http://readwrite.com/2014/02/06/angular-backbone-ember-best-javascript-*Framework*-for-you

Osmani, A. (n.d.). *Developing Backbone.js Applications* (O'Reilly., pp. 1–299). Acedido em http://addyosmani.github.io/backbone-fundamentals/

Osmani, A. (2012). *Learning JavaScript Design Patterns* (p. 1–164). Acedido em http://books.google.com/books?hl=en&lr=&id=L46fX62D5qYC&oi=fnd&pg=PR3&dq=Learning+JavaScript+Design+Patterns&ots=Tmx15lVYgf&sig=uQeOyR-ICanLCBlG1yOrK13rRlY

Osmani, A. (2012a). Review of JS *Framework*s — Journey Through The JavaScript MVC Jungle. *Essentials, Javascript, MVC*, 1–22. Acedido em http://coding.smashingmagazine.com/2012/07/27/journey-through-the-javascript-mvc-jungle/

Osmani, A. (2012b). What Is MVC , Or Rather MV *? When Do You Need A JavaScript MV * *Framework* ?

Osmani, A., & Grüneberger, F. J. (2012). *REAL-TIME COLLABORATION SUPPORT FOR JAVASCRIPT*. *Essentials, Javascript, MVC*. Germany. Acedido em http://coding.smashingmagazine.com/2012/07/27/journey-through-the-javascript-mvc-jungle/

Ousterhout, J. K. (2009). Fiz: A Component *Framework* for Web Applications. *Stanford CSD Technical Report*, 1–14. Acedido em http://www.stanford.edu/~ouster/cgi-bin/papers/fiz.pdf

Pinto, M. I. M. (2007). *Caracterização de soluções de comércio electrónico B2B*.

Porto, S. (2013). A comparison of Angular, Backbone, CanJS and Ember - Sebastian's Blog. Retrieved July 06, 2013, from http://sporto.github.io/blog/2013/04/12/comparison-angular-backbone-can-ember/

Quivy, R., & Campenhoudt, L. Van. (1995). *Manual de Investigacao em Ciencias Sociais* (2ª ed., p. 1–281).

Rodriguez, A. (2008). RESTful Web services : The basics, (November), 1–11.

Runeberg, J. (2013). *TO-DO WITH JAVASCRIPT MV * A study into the differences between Backbone . js and AngularJS*.

Sarraf, T. (2013). Os prós e contras do Design Responsivo para comércio eletrônicoBlog do E-Commerce Brasil. Acedido em https://www.ecommercebrasil.com.br/eblog/2013/10/24/pros-contras-design-responsivo-comercio-eletronico/

Schimtz, D., & Lira, D. (2014). *AngularJS na prática*. *Leanpub* (Leanpub.). Acedido em https://leanpub.com/livro-angularJS/read

Shaked, U. (s.d.). AngularJS vs. Backbone.js vs. Ember.js - Choosing Your JS *Framework*. Acedido em Setembro 23, 2014, em http://www.airpair.com/js/javascript-*Framework*-comparison

Shan, P. (2013). Why AngularJS is generally better in Angular vs Ember vs backbone. Acedido em Setembro 24, 2014, em http://voidcanvas.com/why-angularjs-is-generally-better-than-emberjs-and-backbonejs/

Singh, I., & Palmieri, M. (s.d.). *COMPARISON OF CROSS-PLATFORM MOBILE DEVELOPMENT TOOLS*. *Innovation, Development and Technology*. M¨ alardalen University. Acedido em http://www.idt.mdh.se/kurser/ct3340/ht11/MINICONFERENCE/FinalPapers/ircse11_submission_16.pdf

Teixeira, J. R. (s.d.). Apresentando o *Framework* JavaScript ExtJS. Acedido em http://www.devmedia.com.br/apresentando-o-*Framework*-javascript-extjs/27818

Trager, B., & Kagan, R. (2013). The Battle of Modern Javascript *Framework*s. Acedido em Julho 03, 2013, em http://www.softfinity.com/blog/the-battle-of-modern-javascript-*Framework*s-part-i/

XI. ANEXOS

A. Anexo 1 - Guião de entrevista

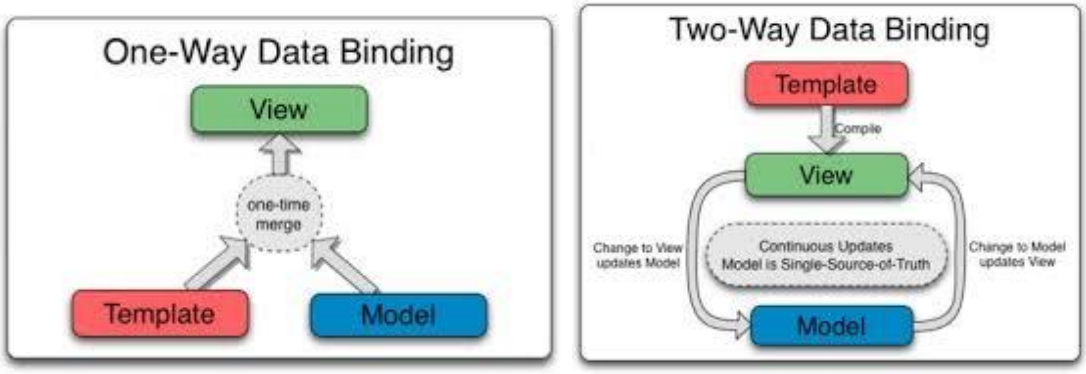
1. Nome, idade, Área de Formação, empresa
2. Quais foram os projetos que desenvolveu até à atualidade? Qual a tipologia dos mesmos?
3. Para que tipo de plataformas se aplicaram alguns desses projetos?
4. Utilizou alguma *Framework* de javascript em alguns desses projetos? Se sim, quais?
5. Qual o motivo principal de optar pelo uso de uma *Framework* no código front end dos seus projetos?
6. Como se deu o processo de escolha, qual a ordem dos passos efetuados?
    a. Essa escolha foi feita pelo programador ou pelo gestor de projeto?
    b. Se pelo gestor, sabe dizer mais ou menos como foi feito o processo?
        i. Quais os critérios utilizados para a escolha da(s) *Framework*(s) nos diferentes projetos?
        ii. Como foram selecionados esses critérios, com base em que requisitos?
        iii. Houveram dificuldades na escolha da *Framework* devido à vasta área de ferramentas atuais?
            i) Quais?
        iv. Utilizou alguma tecnologia ou ferramenta para auxiliar na escolha da *Framework*?
            Se sim, qual?
        v. Qual foi o tempo despendido desde a seleção dos critérios até à decisão final?
7. A escolha foi unanime em todos os projetos?
        a. Se não, quais eram as indecisões?
8. Utiliza ou utilizava mais alguma *Framework* para CSS ou para dispositivos móveis?
    a. Se sim, quais?
    b. A *Framework* escolhida teve alguma influência positiva ou negativa nestas?
        i. Qual?
9. Qual a ordem de decisões entre as ferramentas front-end? Desde CSS até à ligação de dados front-end e back-end.
    a. Acha que essa ordem teve influência na escolha da *Framework*?
10. Fale um pouco da sua experiência com a *Framework*…
11. Quanto tempo, mais ou menos, levou a equipa e você a aprenderem como a *Framework* funciona?
    a. Considera muito ou pouco tempo?
12. Estás satisfeito com a sua decisão?
    a. Quais as maiores vantagens que esta *Framework* lhe oferece?
13. Tem mais algum aspeto que queira acrescentar?

B. Anexo 2 – Resumo da análise dos dados da primeira entrevista

Tópico analisado	Resumo das respostas obtidas
*Framework*s	**Jquery** (3), **Angularjs** (2), KnockoutJs (1), EmberJs (1), Dart (1), Prototype (1), YUI (1), Twin Max(1).
Motivos principais	- Experiência do programador; - Tipologia do projeto; - Evolução do javascript; - Quantidade de código a correr no cliente; - Suporte/Compatibilidade;
Responsável pela escolha	Programador escolhe e gestor de projeto aprova.
Processo	- Tabelas comparativas; - Protótipos funcionais; - Vantagens e desvantagens; - Flexibilidade face à variabilidade do tipo de projeto e necessidades do mesmo; - Gráficos de complexidade; - Estatísticas por comunidade;
Principais critérios	**Comunidade** (3), Documentação (2), **Compatibilidade** (3), **Curva de aprendizagem** (3), Experiência (2), *Plugins* (2), Requisitos do projeto (1), Popularidade (2), *Open source* (1), Atualizada (1), Funcionalidades (1), Tamanho (1), Número de linhas de código (1), Performance (1), Maturidade (1), Dependências (1), Flexibilidade (1).
Dificuldades sentidas	Número alargado de *Frameworks* existentes.
Duração do processo de escolha	Em média uma semana a dois meses, contudo, a maioria das respostas era baseada na urgência de iniciação do projeto.
Escolha unanime entre equipa	Maioria unânime.
*Framework*s CSS e móveis	**Bootstrap** (4), Jquery Mobile (1), Aplicações nativas (1), Foundation (1), Modernizr (1), Less (1).
Problemas de incompatibilidade	Não foram detetados.
Curva de	Influenciada pelo tempo de desenvolvimento do projeto e pelas suas

Tópico analisado	Resumo das respostas obtidas
aprendizagem	necessidades.
Satisfação	Pessoalmente, todos se mostraram satisfeitos.

C. Anexo 3 – One way data binding e two way data binding



D. Anexo 4 – Guião de entrevista aos elementos da empresa

Apresentação

1. Nome

2. Idade

3. Área de formação

4. Papel que desempenha na empresa Prodcent, Lda

Conhecimento da *Framework* Angularjs

5. Já tinha conhecimento da *Framework* antes do projeto iniciar? Se sim, o que conhecia?

6. Caracterize a *Framework* angularjs de acordo com os seguintes critérios**:**

- Comunidade (se a *Framework* é conhecida e dispõe de bastantes fóruns e questões abertas por forma a se obter soluções mais rapidamente**):**

- Curva de aprendizagem (se a *Framework* é fácil e rápida de aprender**):**

- Documentação (se a informação disponível no website é organizada e percetível ao ponto de tornar fácil e acessível a aprendizagem da *Framework*)**:**

- Performance (se a *Framework* se pode tornar problemática a nível de memória e performance para um website, devido às suas funcionalidades**):**

- Popularidade (se a *Framework* é estável e conhecida/utilizada por grandes empresas prestigiadas**):**

7. Quais os aspetos mais positivos e mais negativos que considera acerca da *Framework* Angularjs?

Validade da *Framework* AngularJs para o projeto

8. Quais as principais vantagens que considera que esta *Framework* traz ao projeto em desenvolvimento? E desvantagens?

9. Porque considera esta *Framework* útil para o projeto?

10. Fale um pouco da sua experiência com a *Framework* no projeto a decorrer

11. Tem mais alguma coisa a acrescentar?